Deploying a React Website Using GitHub Actions and Self-Hosted Runners

Yahya Elmokhtari Souhail El Mahdani

May 28, 2024

1 Introduction

This report details the steps taken to deploy a React website using GitHub Actions and a self-hosted runner on a Linux server. The process involved configuring a self-hosted runner on a Linux machine, addressing various issues encountered during deployment, and finally achieving a successful deployment.

2 Setup GitHub Actions and Self-Hosted Runner

2.1 Setting Up Self-Hosted Runner

We first set up a self-hosted runner on a Linux server to use our own machine for deployment tasks. The steps are as follows:

1. Go to your GitHub repository and navigate to `Settings` → `Actions` → `Runners`.
2. Click on `New self-hosted runner`, select `Linux`, and follow the instructions to download and configure the runner.
3. Run the following commands in the terminal to configure the runner:

```
    Download
# Create a folder
$ mkdir actions-runner && cd actions-runner# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.316.1.tar.gz -L https://github.com/actions/runn
$ echo "d62de2400eeeacd195db91e2ff011bfb646cd5d85545e81d8f78c436183e09a8  actions-ru
$ tar xzf ./actions-runner-linux-x64-2.316.1.tar.gz


    Configure
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/YAHYA280/deyployment-ReactApp --token A6V4IXS
$ ./run.sh
```

2.2 Preparing Your Local Hosting Machine

To run a React application, the Linux server must have Node.js and npm installed. Below are the steps and commands to install and configure Node.js and npm on a Linux machine.

1. Updating the Package List:
   First, ensure that your package list is up-to-date by running the following command:

   ```
   sudo apt-get update
   ```

2. Installing Node.js and npm :
   Node.js and npm can be installed from the NodeSource repository. This ensures you get the latest stable version.

(a) Install the NodeSource repository:

```
curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
```

(b) Install Node.js and npm:

```
sudo apt-get install -y nodejs
```

3. Verifying the Installation:
After installation, verify that Node.js and npm are correctly installed by checking their versions:

```
node -v
npm -v
```

With this setup, you will be able to run projects that require Node.js on the hosting machine. We installed Node.js manually to allow us to adjust parameters and use specific versions as needed, avoiding potential version-related errors. It's important to be aware that even small version changes in Node.js or npm can cause errors. Overall, everything should now be configured correctly.

You can find more information at the following link: `https://nodejs.org/en/download/package-manager/all`

2.3 Configuring SSH Access

1. To deploy files to the server, we needed to set up SSH access:

```
sudo apt-get update
sudo apt-get install openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh
```

We also ensured the SSH service was running and configured properly.

2. Generating SSH Keys To securely connect to the server, we generated an SSH key pair. The following commands create a new SSH key pair:

```
# Generate a new SSH key pair
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

During this process, you will be prompted to enter a file path to save the key and a passphrase. For simplicity, you can press Enter to accept the default path and leave the passphrase empty.

The generated keys will be stored in the `/.ssh` directory:

- `/.ssh/id_rsa`: The private key.
- `/.ssh/id_rsa.pub`: The public key.

3. Accessing the SSH Keys To access and copy your public key to the server, use the following command:

```
# Display the public key
cat ~/.ssh/id_rsa.pub
```

Copy the output and add it to the `/.ssh/authorized_keys` file on your server:

```
# On the server, create the .ssh directory if it doesn't exist
mkdir -p ~/.ssh

# Add the public key to the authorized_keys file
echo "your_public_key_contents" >> ~/.ssh/authorized_keys

# Set the correct permissions
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

3 Deploying the React App

3.1 Preparing the React App

We need to ensure that our React app can be built and deployed. We encountered issues with the `react-scripts` not being recognized:

```
npm install react-scripts
```

This resolved the issue and allowed us to proceed with the build process. (This step should be done in the project folder)

3.2 Creating GitHub Secrets

We need to create the necessary GitHub secrets to store sensitive information securely:

- `SSH_PRIVATE_KEY`: The private SSH key for accessing the server.
- `SERVER_USER`: The username for the server.
- `SERVER_IP`: The IP address of the server.

3.3 Writing the GitHub Actions Workflow

The GitHub Actions workflow (`deploy.yml`) was written to build and deploy the React app:

```
on:
  push:
    branches:
      - main  # Change this to your main branch

jobs:
  build-and-deploy:
    runs-on: self-hosted

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '20.13.1' # Specify the Node.js version you want to use

    - name: Install dependencies
      run: npm install

    - name: Build the React app
      run: npm run build

    - name: Deploy to server
      env:
        SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}
        SERVER_USER: ${{ secrets.SERVER_USER }}
        SERVER_IP: ${{ secrets.SERVER_IP }}
      run: |
        mkdir -p ~/.ssh
        echo "${SSH_PRIVATE_KEY}" > ~/.ssh/id_rsa
        chmod 600 ~/.ssh/id_rsa
        ssh-keyscan -H ${{ secrets.SERVER_IP }} >> ~/.ssh/known_hosts
        rsync -avz --delete ./build/ ${{ secrets.SERVER_USER }}@${{ secrets.SERVER_IP }}
```

4 Troubleshooting

1. These are the errors we encountered during the deployment phase. Usually, by following the steps correctly and ensuring that all dependencies are installed and the runtime environment is properly set up, these errors should not appear. However, this information may be helpful if someone encounters similar types of errors:

2. Git Error

   When attempting to push changes, we encountered errors indicating that the remote repository contained work not present locally. This was resolved by pulling the remote changes first:

   ```
   git pull
   git push --u origin main
   ```

3. Resolving Permission Issues During deployment, we encountered permission denied errors due to incorrect ownership of the deployment directory. This was resolved by changing the ownership:

   ```
   sudo chown -R $USER:$USER /var/www/html
   ```

4. Changing User Passwords To ensure SSH access, we needed to reset the password for the user:

   ```
   sudo passwd yahya
   ```

5 Conclusion

By following these steps, we successfully deployed our React website using GitHub Actions and a self-hosted runner on a Linux server. This process involved configuring the runner, setting up SSH access, resolving issues related to permissions, and ensuring our GitHub Actions workflow was correctly set up.

6 Acknowledgments

This report was prepared by Yahya Elmokhtari and Souhail El Mahdani.