

# GitHub Actions CI/CD with AWS Elastic Beanstalk Deployment

---

Author: Oluwaseun Osunsola

LinkedIn: <https://www.linkedin.com/in/oluwaseun-osunsola-95539b175/>

Project GitHub Link: <https://github.com/Oluwaseunoa/github-actions-demo>

This project demonstrates a complete **CI/CD pipeline** for a Node.js application using **GitHub Actions**, **semantic releases**, and **AWS Elastic Beanstalk** for deployment. It includes **99 screenshots** documenting each step of the process.

## Table of Contents

1. [Project Overview](#)
  2. [Prerequisites](#)
  3. [Project Setup](#)
  4. [Local Testing](#)
  5. [CI Pipeline Setup](#)
  6. [Semantic Release Pipeline](#)
  7. [AWS Setup and Elastic Beanstalk Deployment](#)
  8. [Deploy Workflow](#)
  9. [Release and Deployment Verification](#)
  10. [Project Implementation](#)
- 

## Project Overview

This project sets up a Node.js application with:

- **Express.js server**
  - **Mocha** for testing
  - **GitHub Actions CI pipeline** for continuous integration
  - **Semantic release** for automated versioning
  - **Elastic Beanstalk deployment workflow** triggered by releases
- 

## Prerequisites

- Node.js v20 installed locally
- GitHub repository created
- AWS account with **Elastic Beanstalk** access
- IAM user with **AdministratorAccess**
- GitHub repository secrets for AWS credentials:

- AWS\_ACCESS\_KEY\_ID
  - AWS\_SECRET\_ACCESS\_KEY
  - AWS\_REGION
- 

## Project Setup

1. **Create repository** and initialize with README
  2. **Clone the repository** locally
  3. **Initialize Node.js project** with `npm init -y`
  4. **Install Express:** `npm install express`
  5. **Create app.js** and add basic server code
  6. **Install Mocha for testing:** `npm install --save-dev mocha`
- 

## Local Testing

- Write simple test in `test.js`
  - Update `package.json` scripts for testing
  - Run `npm test` locally to verify functionality
  - Start the server locally: `npm start`
  - Visit `http://localhost:3000` to verify "Hello, World! This is version 1.0.0"
- 

## CI Pipeline Setup

- Create `.github/workflows/ci.yml`
  - Steps include:
    - Checkout code
    - Install dependencies
    - Run tests
  - Push to GitHub triggers **CI workflow**
  - Verify workflow runs successfully
- 

## Semantic Release Pipeline

- Install dependencies for **semantic-release**
  - Configure `.releaserc.json`
  - Add `release.yml` workflow in GitHub Actions
  - Commit conventional commits to trigger **automated releases**
  - Releases tab shows auto-generated release notes
- 

## AWS Setup and Elastic Beanstalk Deployment

- Log into AWS console

- Create IAM user and generate access keys
  - Add repository secrets for AWS credentials
  - Create Elastic Beanstalk application and environment:
    - Node.js platform
    - Single-instance environment
    - Default VPC and public IP
    - EC2 instance profile and service role
  - Configure optional settings for monitoring, rolling updates, logging, and platform options
- 

## Deploy Workflow

- Create `deploy.yml` workflow triggered by **release event**
  - Workflow steps:
    - Checkout code
    - Install dependencies
    - Run tests
    - Zip application
    - Configure AWS credentials
    - Deploy to Elastic Beanstalk using [einaregilsson/beanstalk-deploy](#)
- 

## Release and Deployment Verification

- Create release on GitHub ([v1.3.0](#))
  - Trigger deployment workflow
  - Deployment pipeline runs successfully
  - Visit Elastic Beanstalk URL to confirm application is live
- 

## Project Implementation

---

### Step 1: Create a GitHub Repository

Create a new repository named `github-actions-demo` and initialize it with a README.

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

1 General

Owner \* [Oluwaseunoa](#) / Repository name \*  [github-actions-demo is available.](#)

Description

2 Configuration

Choose visibility \* [Public](#)

Add README [About READMEs](#) [On](#)

Add .gitignore [About ignoring files](#) [No .gitignore](#)

Add license [About licenses](#) [No license](#)

[Create repository](#)

## Step 2: Copy Repository HTTPS URL

Copy the repository HTTPS URL to clone it locally.

https://github.com/Oluwaseunoa/github-actions-demo

Oluwaseunoa / [github-actions-demo](#)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

github-actions-demo Public

main · 1 Branch · 0 Tags

Go to file Add file < > Code Local Codespaces

Clone HTTPS SSH GitHub CLI

<https://github.com/Oluwaseunoa/github-actions-demo>

Clone using the web URL Open with GitHub Desktop Download ZIP

About

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

## Step 3: Clone Repository Locally

Clone the repository using the copied link and navigate into it:

```
git clone <repo-https-url>
cd github-actions-demo
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects (main)
$ git clone https://github.com/Oluwaseunoa/github-actions-demo.git
Cloning into 'github-actions-demo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects (main)
$ cd github-actions-demo

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 4: Verify Repository Contents

List files to verify that `README.md` exists:

```
ls
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ ls
README.md

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 5: Initialize NPM

Initialize the project as a Node.js project:

```
npm init -y
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm init -y
Write to C:\Users\HP\Documents\Workspace\DevOps-Projects\GitHub-Actions-Projects\github-actions-demo\package.json:

{
  "name": "github-actions-demo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Oluwaseunoa/github-actions-demo.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "bugs": {
    "url": "https://github.com/Oluwaseunoa/github-actions-demo/issues"
  },
  "homepage": "https://github.com/Oluwaseunoa/github-actions-demo#readme"
}

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 6: Verify `package.json`

Check that `package.json` now exists in the project folder:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ ls
package.json README.md

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 7: Install Express

Install Express framework as a dependency:

```
npm install express
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm install express

added 65 packages, and audited 66 packages in 5s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 8: Create `app.js` and Open in VS Code

Create the main application file `app.js` and open the project in VS Code:

```
touch app.js
code .
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch app.js

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ code .
```

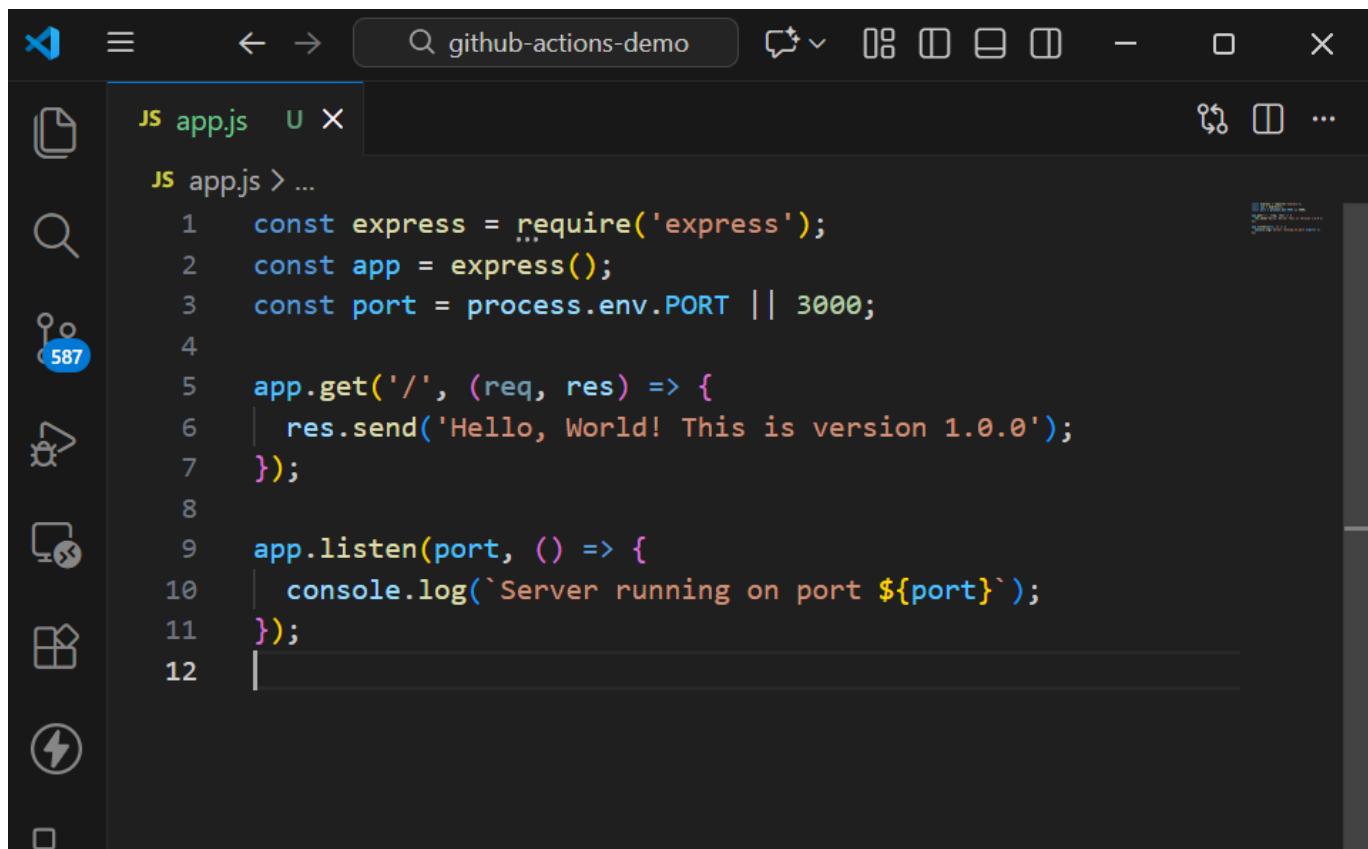
## Step 9: Add Code to `app.js`

Write the initial Node.js server code in `app.js`:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => res.send('Hello World!'));

app.listen(port, () => console.log(`Server running on port ${port}`));
```



The screenshot shows a dark-themed instance of Visual Studio Code. In the center, there's a code editor window titled "app.js". The file contains the following code:

```
1 const express = require('express');
2 const app = express();
3 const port = process.env.PORT || 3000;
4
5 app.get('/', (req, res) => {
6   res.send('Hello, World! This is version 1.0.0');
7 });
8
9 app.listen(port, () => {
10   console.log(`Server running on port ${port}`);
11});
```

The left sidebar features a vertical toolbar with various icons: a file icon, a search icon, a refresh icon with a count of "587", a play icon, a refresh icon, a lightning bolt icon, and a square icon.

---

## Step 10: Install Mocha as Dev Dependency

Install Mocha for testing as a development dependency:

```
npm install --save-dev mocha
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo$
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm install --save-dev mocha

added 90 packages, and audited 156 packages in 7s

51 packages are looking for funding
  run `npm fund` for details

2 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 11: Create Test File

Create a test file named `test.js` inside the project:

```
touch test.js
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch test.js

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 12: Write a Simple Test

Add a simple test in `test.js` to verify basic functionality:

```
const assert = require('assert');

describe('Basic Test', () => {
  it('should return true', () => {
    assert.strictEqual(true, true);
```

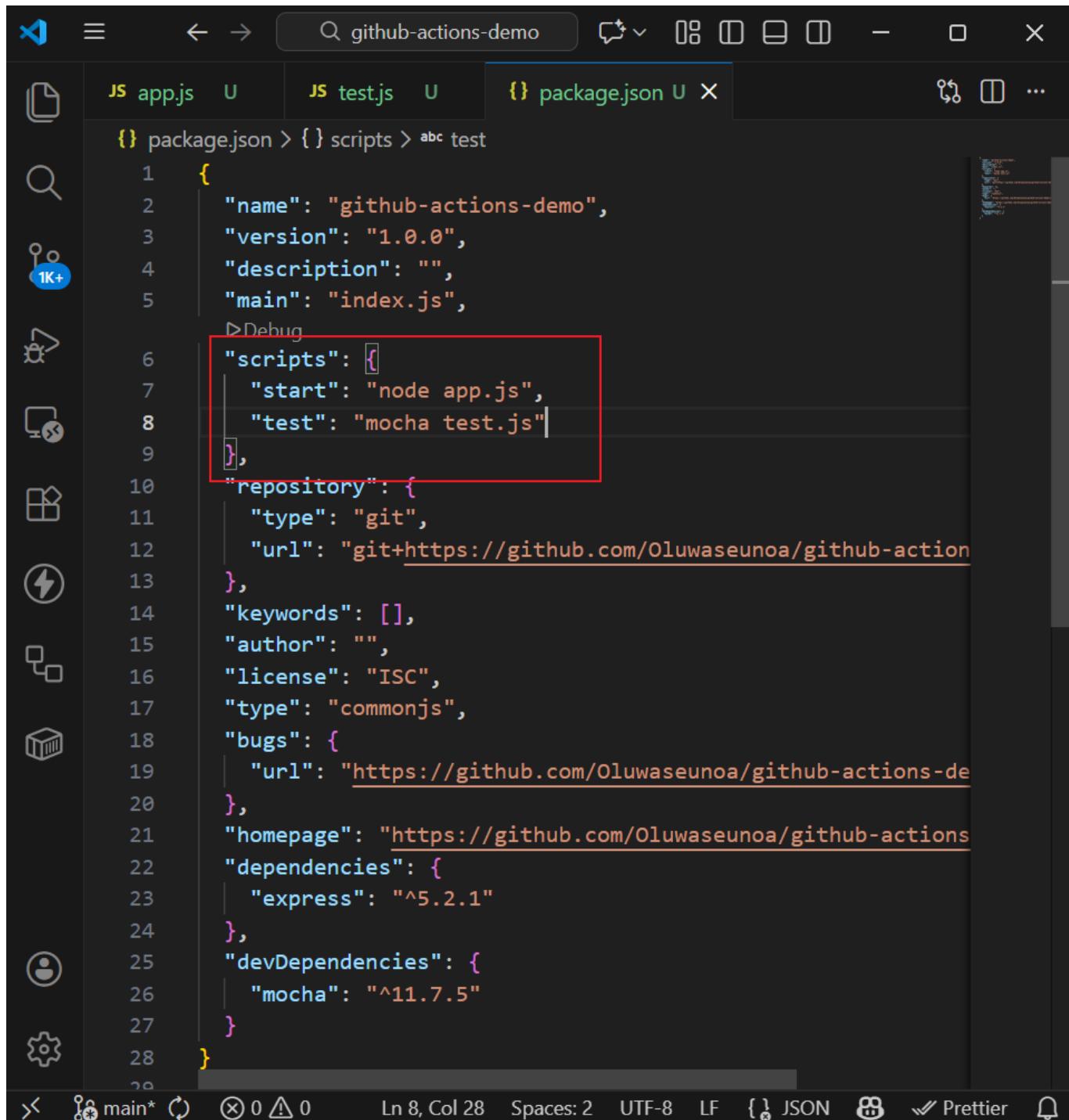
```
});  
});
```



## Step 13: Modify Scripts in `package.json`

Update the `scripts` section in `package.json` to include test and start commands:

```
"scripts": {  
  "test": "mocha",  
  "start": "node app.js"  
}
```



The screenshot shows the VS Code interface with the file `package.json` open. The `scripts` section is highlighted with a red box, indicating it is the focus of the current step. The code in the `scripts` section is:

```
1  {
2    "name": "github-actions-demo",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": [
7      "start": "node app.js",
8      "test": "mocha test.js"
9    ],
10   "repository": {
11     "type": "git",
12     "url": "git+https://github.com/Oluwaseunoa/github-action"
13   },
14   "keywords": [],
15   "author": "",
16   "license": "ISC",
17   "type": "commonjs",
18   "bugs": {
19     "url": "https://github.com/Oluwaseunoa/github-actions-de"
20   },
21   "homepage": "https://github.com/Oluwaseunoa/github-actions"
22   "dependencies": {
23     "express": "^5.2.1"
24   },
25   "devDependencies": {
26     "mocha": "^11.7.5"
27   }
28 }
```

## Step 14: Run Tests Locally

Run tests to ensure they pass:

```
npm test
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo — □ ×

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm test

> github-actions-demo@1.0.0 test
> mocha test.js

Basic Test
  ✓ should return true

1 passing (4ms)

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 15: Run Application Locally

Start the application to test it in a local environment:

```
npm run start
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm start

> github-actions-demo@1.0.0 start
> node app.js

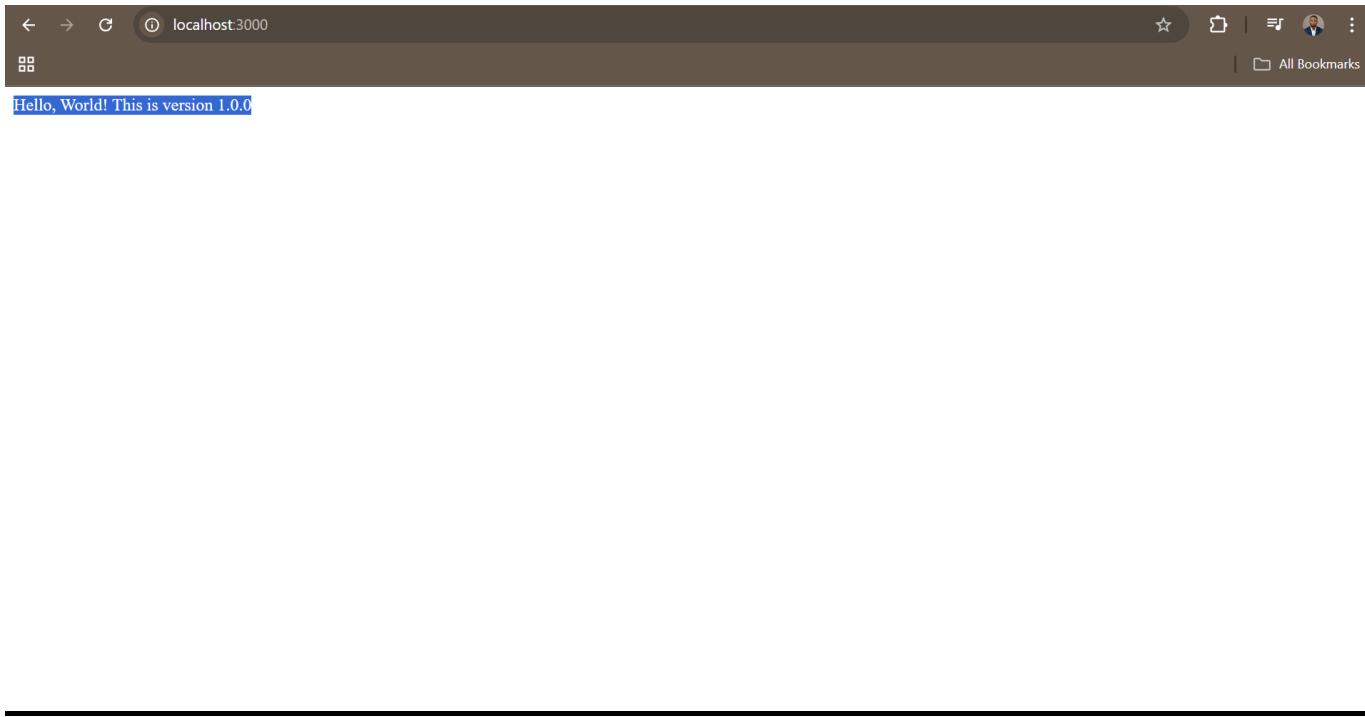
Server running on port 3000
|
```

---

## Step 16: Verify Application in Browser

Visit <http://localhost:3000> to check the running application. It should display:

```
Hello, World! This is version 1.0.0
```



---

## Step 17: Stop the Server

Press **Ctrl + C** in the terminal to stop the running server:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm start

> github-actions-demo@1.0.0 start
> node app.js

Server running on port 3000

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 18: Create `.gitignore` File

Create a `.gitignore` file to exclude unnecessary files from Git:

```
touch .gitignore
```

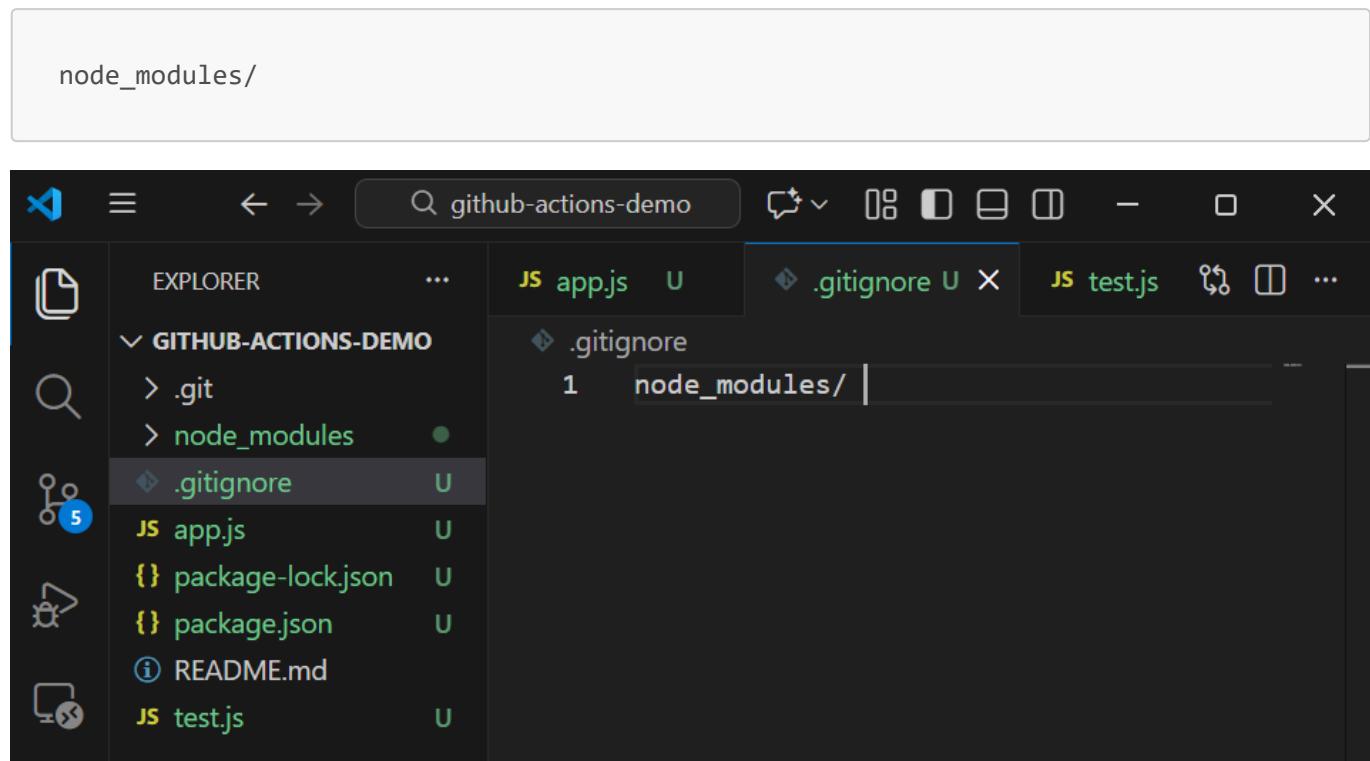
```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch .gitignore
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 19: Add `node_modules` to `.gitignore`

Prevent committing `node_modules` by adding it to `.gitignore`:

```
node_modules/
```



The screenshot shows the VS Code interface with the search bar set to "github-actions-demo". The Explorer sidebar shows a folder named "GITHUB-ACTIONS-DEMO" containing files like ".git", "node\_modules", ".gitignore", "app.js", "package-lock.json", "package.json", "README.md", and "test.js". The ".gitignore" file is selected in the Explorer. The code editor shows the content of the ".gitignore" file, which contains a single line: "node\_modules/".

## Step 20: Commit and Push Changes

Commit all changes and push to the remote repository:

```
git add .
git commit -m "setup initial project structure, tests, and server"
git push origin main
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "Initial Node.js app setup"
[main 7a32d3d] Initial Node.js app setup
5 files changed, 2023 insertions(+)
create mode 100644 .gitignore
create mode 100644 app.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 test.js
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 17.11 KiB | 5.70 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/OluwaseunO/greenlight-demos.git
   cd/b59a..7a32d3d  main -> main
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

Perfect! Let's continue the README with **steps 21–30**. I'll maintain the same professional format and include all screenshots.

## Step 21: Verify Commit on GitHub

Check that your recent commit is now reflected on GitHub:

The screenshot shows a GitHub repository page for 'github-actions-demo'. The repository is public and has 2 commits. The commits are:

- Oluwaseunoa Initial Node.js app setup (7a32d3d · 9 minutes ago)
- .gitignore Initial Node.js app setup (9 minutes ago)
- README.md Initial commit (1 hour ago)
- app.js Initial Node.js app setup (9 minutes ago)
- package-lock.json Initial Node.js app setup (9 minutes ago)
- package.json Initial Node.js app setup (9 minutes ago)
- test.js Initial Node.js app setup (9 minutes ago)

The repository has 0 stars, 0 forks, and 0 releases. It uses JavaScript 100.0%.

## Step 22: Create Workflow Directory

Create a `.github/workflows` directory in your project to store GitHub Actions workflows:

```
mkdir -p .github/workflows
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ mkdir -p .github/workflows
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 23: Create `ci.yml` Workflow File

Inside the workflows directory, create a `ci.yml` file for CI configuration:

```
touch .github/workflows/ci.yml
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch .github/workflows/ci.yml
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 24: Add Pipeline Script to `ci.yml`

Edit `ci.yml` to include the pipeline configuration, e.g., for testing Node.js application:

```
name: Node.js CI
on:
  push:
    branches: [ main ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install
      - run: npm test
```

The screenshot shows the VS Code interface with the repository 'github-actions-demo' open. The file 'ci.yml' is the active tab, displaying a GitHub Workflow YAML configuration. The code is as follows:

```
.github > workflows > ! ci.yml > {} on > {} pull_request > [] branches
GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
1 name: CI Pipeline
2
3 on:
4   pull_request:
5     branches:
6       - main
7
8 jobs:
9   test:
10    name: Run Tests
11    runs-on: ubuntu-latest
12
13 steps:
14   - name: Checkout repository
15     uses: actions/checkout@v4
16
17   - name: Set up Node.js
18     uses: actions/setup-node@v4
19     with:
20       node-version: '20'
21
22   - name: Install dependencies
23     run: npm install
24
25   - name: Run unit tests
26     run: npm test
27
```

The status bar at the bottom shows the file is main\*, has 0 changes, and is in UTF-8 CRLF mode. It also indicates the file is a YAML file and part of a GitHub Workflow, with Prettier being used.

## Step 24b: Commit and Push Workflow File

Add, commit, and push the newly created workflow file to the repository:

```
git add .github/workflows/ci.yml
git commit -m "add CI workflow file"
git push origin main
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .github/workflows/ci.yml

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "ci: add Github Actions pipeline"
[main 845c2d9] ci: add Github Actions pipeline
 1 file changed, 26 insertions(+)
 create mode 100644 .github/workflows/ci.yml

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 602 bytes | 200.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Oluwaseunoa/github-actions-demo.git
 7a32d3d..845c2d9 main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 25: Create a Feature Branch

Create a new feature branch for updates:

```
git checkout -b feature/ci-test
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git checkout -b feature/ci-test
Switched to a new branch 'feature/ci-test'

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$
```

---

## Step 26: Edit `app.js` and Send to Hello, World! CI Pipeline Active

Make edits in `app.js` and ensure that the CI pipeline is active and triggered:

```

    JS app.js M X .gitignore ci.yml test.js package.json
    JS app.js > ...
    1 const express = require('express');
    2 const app = express();
    3 const port = process.env.PORT || 3000;
    4
    5 app.get('/', (req, res) => {
    6   | res.send('Hello, World! CI Pipeline Active 🚀');
    7 });
    8
    9 app.listen(port, () => {
    10  | console.log(`Server running on port ${port}`);
    11 });
    12
  
```

Oluwaseun Osunsola (23 minutes ago) Ln 7, Col 4 Spaces: 4 UTF-8 CRLF C JavaScript Prettier

## Step 27: Commit and Push Feature Branch

Commit changes to the feature branch and push to GitHub:

```

git add .
git commit -m "update app.js for CI test"
git push origin feature/ci-test
  
```

```

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$ git add app.js

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$ git commit -m "test: verify CI pipeline"
[feature/ci-test f47482d] test: verify CI pipeline
 1 file changed, 1 insertion(+), 1 deletion(-)

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$ git push origin feature/ci-test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/ci-test' on GitHub by visiting:
remote:     https://github.com/Oluwaseunoa/github-actions-demo/pull/new/feature/ci-test
remote:
To https://github.com/Oluwaseunoa/github-actions-demo.git
 * [new branch]      feature/ci-test -> feature/ci-test

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$ 
  
```

## Step 28: Open Pull Request on GitHub

Go to GitHub, compare your feature branch to [main](#), and ensure your changes are ready for a Pull Request:

The screenshot shows a GitHub repository page for 'github-actions-demo'. A red arrow points to the 'Compare & pull request' button in the top right corner of the main content area.

## Step 29: Create Pull Request

Scroll down on GitHub and click **Create Pull Request** to merge the feature branch:

The screenshot shows the 'Compare & pull request' view on GitHub. A red arrow points to the 'Create pull request' button at the bottom of the page.

## Step 30: CI Pipeline Triggered

Once the Pull Request is created, the CI pipeline is automatically triggered and should pass:

test: verify CI pipeline #1

Oluwaseunoa wants to merge 1 commit into `main` from `feature/ci-test`

Conversation 0 Commits 1 Checks 0 Files changed 1

Oluwaseunoa commented now  
No description provided.

test: verify CI pipeline

All checks have passed  
1 successful check

CI Pipeline / Run Tests (pull\_request) Successful in 12s

No conflicts with base branch  
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions](#).

Reviewers  
No reviews  
Still in progress? Convert to draft

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Development  
Successfully merging this pull request may close these issues.  
None yet

Great! Let's continue the README with **steps 31–40**, keeping it clean, professional, and consistent with the previous sections.

## Step 31: Merge Pull Request

Click **Merge Pull Request** on GitHub to merge your feature branch into `main`.

test: verify CI pipeline #1

Oluwaseunoa wants to merge 1 commit into `main` from `feature/ci-test`

Conversation 0 Commits 1 Checks 0 Files changed 1

Oluwaseunoa commented 2 minutes ago  
No description provided.

test: verify CI pipeline

All checks have passed  
1 successful check

CI Pipeline / Run Tests (pull\_request) Successful in 12s

No conflicts with base branch  
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions](#).

Reviewers  
No reviews  
Still in progress? Convert to draft

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Development  
Successfully merging this pull request may close these issues.  
None yet

## Step 32: Confirm Merge

Confirm the merge to complete integrating the feature branch:

A screenshot of a GitHub pull request merge dialog. The URL is [github.com/Oluwaseunoa/github-actions-demo/pull/1](https://github.com/Oluwaseunoa/github-actions-demo/pull/1). The title is "test: verify CI pipeline #1". The "Conversation" tab shows one message from "Oluwaseunoa" 3 minutes ago: "No description provided.". The "Commits" tab shows 1 commit: "test: verify CI pipeline" by "Oluwaseunoa" (f47482d). The "Checks" tab shows 0 checks. The "Files changed" tab shows 1 file. On the right, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (Successfully merging this pull request may close these issues, None yet). The "Notifications" section shows "Unsubscribe" and "Customize". A red arrow points to the "Confirm merge" button at the bottom left of the dialog.

## Step 33: Update Local Main Branch

Checkout to the `main` branch locally and pull the latest changes from GitHub:

```
git checkout main
git pull origin main
```

A screenshot of a terminal window titled "MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo". The command history shows:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (feature/ci-test)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 906 bytes | 100.00 KiB/s, done.
From https://github.com/Oluwaseunoa/github-actions-demo
  845c2d9..3c95305  main      -> origin/main
Updating 845c2d9..3c95305
Fast-forward
 app.js | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 34: Install Semantic Release Dependencies

Install dependencies needed for automated semantic releases:

```
npm install --save-dev semantic-release @semantic-release/npm @semantic-release/github @semantic-release/commit-analyzer @semantic-release/release-notes-generator
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ npm config set fetch-retry-mintimeout 20000
npm config set fetch-retry-maxtimeout 120000
npm install --save-dev semantic-release @semantic-release/commit-analyzer @semantic-release/release-notes-generator @semantic-release/npm @semantic-release/github
npm warn deprecated semver-diff@5.0.0: Deprecated as the semver package now supports this built-in.

added 248 packages, and audited 612 packages in 3m

137 packages are looking for funding
  run `npm fund` for details

  14 vulnerabilities (2 low, 12 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 35: Create .releaserc.json

Create a `.releaserc.json` file in the root directory to configure semantic release:

```
touch .releaserc.json
```

```
☰ MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo □ ×

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch .releaserc.json

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 36: Add Code to .releaserc.json

Add the following configuration to `.releaserc.json`:

```
{
  "branches": ["main"],
  "plugins": [
    "@semantic-release/commit-analyzer",
    "@semantic-release/release-notes-generator",
    "@semantic-release/npm",
    "@semantic-release/github"
  ]
}
```

The screenshot shows the VS Code interface with the project 'github-actions-demo' open. The Explorer sidebar shows files like .git, .github, node\_modules, .gitignore, .releaserc.json, app.js, package-lock.json, package.json, README.md, and test.js. The .releaserc.json file is selected and its content is displayed in the main editor:

```
1 {  
2   "branches": ["main"],  
3   "plugins": [  
4     "@semantic-release/commit-analyzer",  
5     "@semantic-release/release-notes-generator",  
6     "@semantic-release/npm",  
7     "@semantic-release/github"  
8   ]  
9 }  
10 }
```

## Step 37: Prevent Publishing to NPM

Add `"private": true` to `package.json` to prevent accidental publishing to NPM:

```
"private": true
```

The screenshot shows the VS Code interface with the project 'github-actions-demo' open. The Explorer sidebar shows files like .git, .github, node\_modules, .gitignore, .releaserc.json, app.js, package-lock.json, package.json, README.md, and test.js. The package.json file is selected and its content is displayed in the main editor. A red arrow points to the `"private": true` line, which is highlighted.

```
1 {  
2   "name": "github-actions-demo",  
3   "version": "1.0.0",  
4   "private": true, // Red arrow points here  
5   "description": "",  
6   "main": "index.js",  
7   "scripts": {  
8     "start": "node app.js",  
9     "test": "mocha test.js"  
10    },  
11   "repository": {
```

## Step 38: Create `release.yml` Workflow

Create a new GitHub Actions workflow file `release.yml` in `.github/workflows`:

```
touch .github/workflows/release.yml
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch .github/workflows/release.yml
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 39: Add Release Pipeline Script

Edit `release.yml` to include the semantic release deployment pipeline:

```
name: Release Pipeline
on:
  push:
    branches:
      - main
jobs:
  release:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm ci
      - run: npx semantic-release
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      NPM_TOKEN: ${{ secrets.NPM_TOKEN }}
```

The screenshot shows the GitHub Actions release.yml configuration file in a code editor. The file defines a workflow named 'Release Pipeline' that triggers on pushes to the 'main' branch. It includes steps for checking out the repository, setting up Node.js (version 20), and installing dependencies.

```
.github > workflows > ! release.yml > ...
GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
1 name: Release Pipeline
2
3 permissions:
4   contents: write
5   issues: write
6   pull-requests: write
7
8 on:
9   push:
10    branches:
11      - main
12
13 jobs:
14   release:
15     name: Semantic Release
16     runs-on: ubuntu-latest
17
18   steps:
19     - name: Checkout repository
20       uses: actions/checkout@v4
21       with:
22         fetch-depth: 0
23
24     - name: Set up Node.js
25       uses: actions/setup-node@v4
26       with:
27         node-version: '20'
28
29
30 name: Install dependencies
```

---

## Step 39b: Configure GitHub Actions Permissions

Go to **Repository** → **Settings** → **Actions** → **General** and enable **Read & Write** permissions for workflows, and allow GitHub Actions to create and approve Pull Requests. Then save the settings.

The screenshot shows the GitHub Actions settings page for a repository. On the left, there's a sidebar with 'Security' (Advanced Security, Deploy keys, Secrets and variables), 'Integrations' (GitHub Apps, Email notifications), and a 'Save' button. The main area has a heading 'Approval for running fork pull request workflows from contributors' with three radio button options: 'Require approval for first-time contributors who are new to GitHub' (selected), 'Require approval for first-time contributors' (Only users who have never had a commit or pull request merged into this repository will require approval to run workflows.), and 'Require approval for all external contributors' (All users that are not a member or owner of this repository will require approval to run workflows.). Below this is a 'Workflow permissions' section with two radio button options: 'Read and write permissions' (Workflows have read and write permissions in the repository for all scopes) and 'Read repository contents and packages permissions' (Workflows have read permissions in the repository for the contents and packages scopes only). A red box highlights the 'Allow GitHub Actions to create and approve pull requests' checkbox, which is checked. A red arrow points to the 'Save' button at the bottom.

## Step 40: Commit Conventional Commit and Push

Commit all changes using a conventional commit message and push to GitHub:

```
git add .
git commit -m "chore: configure semantic release"
git push origin main
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "chore(release): configure semantic release"
[main 5b97f5a] chore(release): configure semantic release
 4 files changed, 6034 insertions(+), 160 deletions(-)
  create mode 100644 .github/workflows/release.yml
  create mode 100644 .releaserc.json

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 50.09 KiB | 2.64 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/oluwaseunoa/github-actions-demo.git
 3c95305..5b97f5a main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

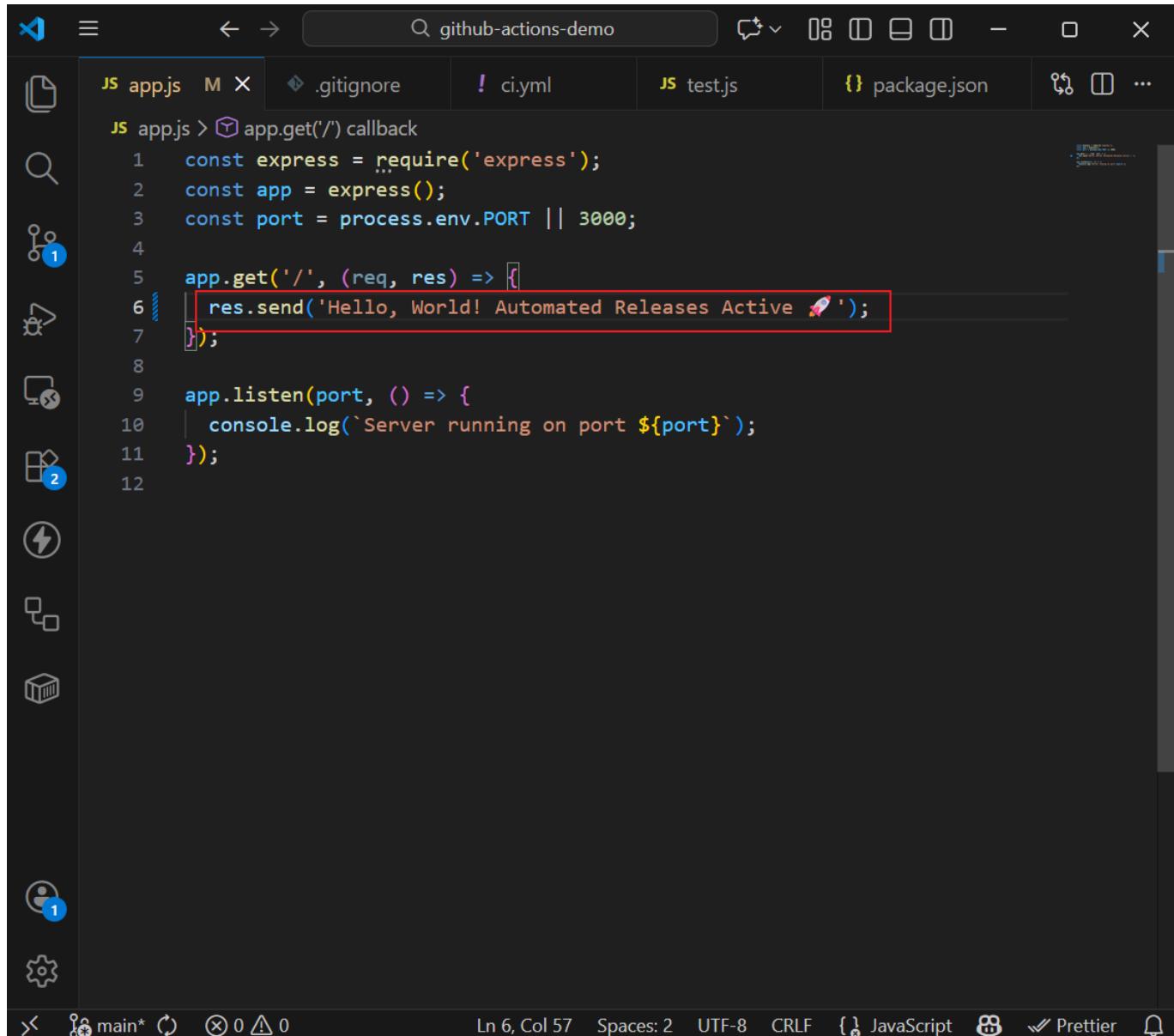
---

Perfect! Let's continue the README with **steps 41–50**, keeping the professional style and linking each screenshot.

---

## Step 41: Edit `app.js` to Trigger Pipeline

Make the necessary edits in `app.js` that will trigger the release pipeline upon commit:



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files: `app.js`, `.gitignore`, `ci.yml`, `test.js`, and `package.json`.
- Editor:** The `app.js` file is open, showing the following code:

```
1 const express = require('express');
2 const app = express();
3 const port = process.env.PORT || 3000;
4
5 app.get('/', (req, res) => [
6   |   res.send('Hello, World! Automated Releases Active 🚀');
7 ]);
8
9 app.listen(port, () => {
10   console.log(`Server running on port ${port}`);
11});
```

A red box highlights the line `res.send('Hello, World! Automated Releases Active 🚀');`.
- Sidebar:** Shows notifications (1), a diff view (2), and other icons.
- Bottom:** Status bar shows: `main*`, `Ln 6, Col 57`, `Spaces: 2`, `UTF-8`, `CRLF`, `JavaScript`, `Prettier`, and a clipboard icon.

---

## Step 42: Commit and Push Changes

Commit your edits and push them to GitHub:

```
git add .
git commit -m "fix: trigger pipeline for testing release"
git push origin main
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add app.js

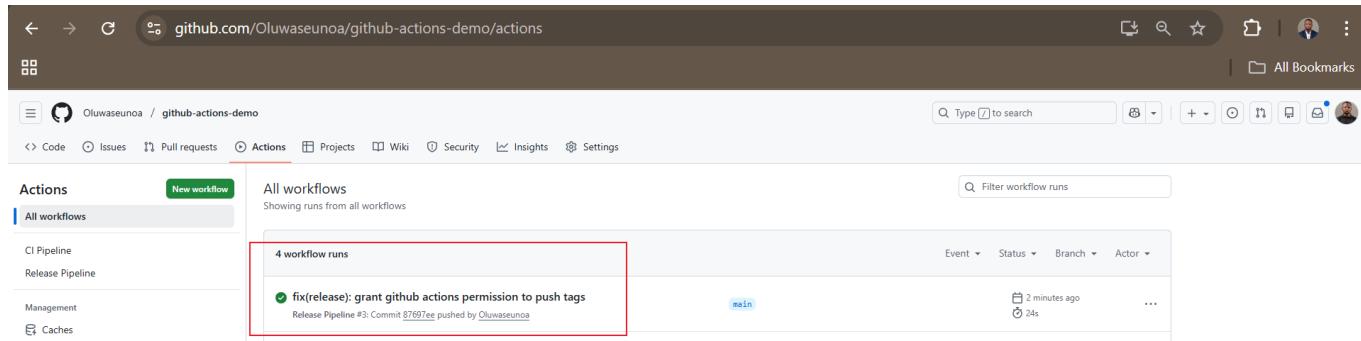
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "feat: enable automated semantic releases"
[main 1811cf8] feat: enable automated semantic releases
 1 file changed, 1 insertion(+), 1 deletion(-)

MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 331 bytes | 165.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Oluwaseunoa/github-actions-demo.git
  5b97f5a..1811cf8  main -> main

MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 43: Verify Workflow Run

Go to the **Actions** tab in GitHub to ensure the workflow run completed successfully with no errors:

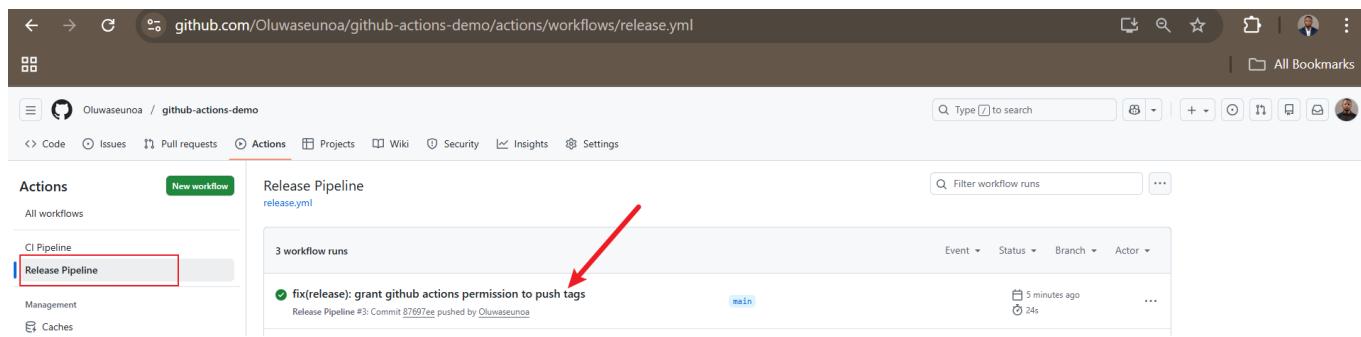


The screenshot shows the GitHub Actions tab for the repository 'Oluwaseunoa/github-actions-demo'. The 'Actions' tab is selected. A single workflow run is listed under 'All workflows':

- fix(release): grant github actions permission to push tags**
- Event: main
- Status: 2 minutes ago
- Actor: Oluwaseunoa
- Branch: main
- Duration: 24s

## Step 44: Access Release Pipeline

Click on the **Release Pipeline** to view the detailed run:



The screenshot shows the GitHub Actions tab for the repository 'Oluwaseunoa/github-actions-demo'. The 'Actions' tab is selected. A single workflow run is listed under 'All workflows' for the 'Release Pipeline' workflow:

- fix(release): grant github actions permission to push tags**
- Event: main
- Status: 5 minutes ago
- Actor: Oluwaseunoa
- Branch: main
- Duration: 24s

## Step 45: Release Pipeline Completed

Check that the release pipeline has run completely and all steps succeeded:

The screenshot shows the GitHub Actions pipeline details for a job named "fix(release): grant github actions permission to push tags #3". The job status is green, indicating success. The job name is "Semantic Release". The pipeline summary shows the following steps:

- > Set up job (1s)
- > Checkout repository (1s)
- > Set up Node.js (0s)
- > Install dependencies (6s)
- > Run Semantic Release (9s)
- > Post Set up Node.js (0s)
- > Post Checkout repository (0s)
- > Complete job (0s)

## Step 46: Verify Release Version on GitHub

On the repository dashboard, the new release version now displays:

The screenshot shows the GitHub repository dashboard for "github-actions-demo". The repository owner is "Oluwaseunoa". The repository name is "github-actions-demo". The repository page shows the following details:

- main branch (2 Branches, 1 Tag)
- Activity: 8 Commits
- Releases: 1 (v1.0.0 [Latest] 8 minutes ago)
- Packages: No packages published
- Languages: JavaScript 100.0%

## Step 47: View Release Details

Clicking on the release version shows the release notes and details of the fixes included:

The screenshot shows the GitHub release page for version 1.0.0. A red box highlights the "Bug Fixes" section, which contains a single item: "release: grant github actions permission to push tags (87697ee)". Below this is the "Features" section, which also contains one item: "enable automated semantic releases (1811cf8)".

## Step 48: Receive Email Notification

If configured, receive an email notification about the new bug fixes and release:

The screenshot shows an email in the Gmail inbox. The subject is "Re: [Oluwaseunoa/github-actions-demo] test: verify CI pipeline (PR #1) [inbox x]". The email is from "github-actions[bot] <notifications@github.com>" and was sent "12 minutes ago". The message body includes a comment from "github-actions[bot]" linking to the GitHub release and noting that the PR is included in version 1.0.0.

## Step 49: Log into AWS Console

Log into the AWS Management Console and navigate to **IAM** to manage users and permissions:

The screenshot shows the AWS IAM dashboard. On the left sidebar, under the 'Services' section, the 'IAM' icon is highlighted with a red arrow. The main content area displays sections for 'Applications', 'Groups', 'Roles', and 'Resources'. A modal window titled 'Looking for resources in other Regions?' is open, providing instructions on how to enable cross-Region search.

## Step 50: Access Users in IAM

Click on **Users** in the IAM dashboard to see the list of IAM users:

The screenshot shows the AWS IAM Dashboard. The left sidebar has a 'Users' link under the 'Access Management' section, which is highlighted with a red arrow. The main content area includes sections for 'Security recommendations', 'IAM resources' (showing 1 user, 8 roles, 0 policies, 0 identity providers), 'What's new', and 'AWS Account' information. The 'Quick Links' section contains a 'My security credentials' link.

## Step 51: Verify or Create Active Access Key

Ensure the IAM user has an **active access key**. If not, create one. Note the displayed key for use in GitHub Secrets:

The screenshot shows the AWS IAM User Details page for a user named 'seun-testing'. The 'Permissions' tab is active. In the 'Security credentials' section, there is one access key listed. A red box highlights this section, showing the Access Key ID 'AKIA4D4CIJ2Y5TDJA540' and a note indicating it was used 26 days ago, 33 days old.

## Step 52: Copy AWS Credentials

On your local machine, view your AWS credentials and copy the **Access Key ID** and **Secret Access Key**:

```
cat ~/.aws/credentials
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ cat ~/.aws/credentials

[testing]
aws_access_key_id = AKIAZ5NF3I73AUYU4EW
aws_secret_access_key = [REDACTED]

[testing1]
aws_access_key_id = AKIA4D4CIJ2Y5TODNGHV
aws_secret_access_key = [REDACTED]

[seun-testing]
aws_access_key_id = AKIA4D4CIJ2Y5TDJA540
aws_secret_access_key = [REDACTED] !aX8vDac9Aj6aJQ

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 53: Open GitHub Secrets

Go to your repository on GitHub, click **Settings** → **Secrets and variables** → **Actions**:



## Step 54: Add New Repository Secret

Click **New repository secret** to add AWS credentials:

The screenshot shows the GitHub Actions settings page for a repository. The left sidebar has sections like General, Access, Collaborators, Moderation options, Code and automation, Security, Advanced Security, Deploy keys, Secrets and variables, and Actions. The 'Secrets and variables' section is expanded. The main area is titled 'Actions secrets and variables'. It contains two tabs: 'Secrets' (selected) and 'Variables'. Under 'Secrets', there's a section for 'Environment secrets' which says 'This environment has no secrets.' and a 'Manage environment secrets' button. Below it is a section for 'Repository secrets' which also says 'This repository has no secrets.' and a green 'New repository secret' button. A red arrow points to this 'New repository secret' button.

## Step 55: Add AWS\_ACCESS\_KEY\_ID

Paste **AWS\_ACCESS\_KEY\_ID** as the **name** and the Access Key ID as the **secret value**, then click **Add secret**:

The screenshot shows the 'New secret' creation page for GitHub Actions. The left sidebar is identical to the previous screenshot. The main area is titled 'Actions secrets / New secret'. It has fields for 'Name \*' (containing 'AWS\_ACCESS\_KEY\_ID') and 'Secret \*' (containing 'AKIA4D4CII2Y5TDJA54O'). At the bottom right is a green 'Add secret' button. A red arrow points to this 'Add secret' button.

## Step 56: Add AWS\_SECRET\_ACCESS\_KEY

Similarly, add **AWS\_SECRET\_ACCESS\_KEY** with its value and save as a secret:

The screenshot shows the 'Actions secrets / New secret' page on GitHub. The 'Name' field is filled with 'AWS\_SECRET\_ACCESS\_KEY'. The 'Secret' field contains a long, obscured string of characters. A red arrow points to the green 'Add secret' button at the bottom right of the secret input area.

## Step 57: Add AWS\_REGION

Finally, add **AWS\_REGION** as a secret with the appropriate AWS region value (e.g., **us-east-1**):

The screenshot shows the 'Actions secrets / New secret' page on GitHub. The 'Name' field is filled with 'AWS\_REGION'. The 'Secret' field contains 'us-east-2'. A red arrow points to the green 'Add secret' button at the bottom right of the secret input area.

## Step 58: Open Elastic Beanstalk Console

In the AWS Console, search for **Elastic Beanstalk** and click on it to start deploying your application:

The screenshot shows the AWS IAM console with a search bar at the top containing 'Elastic Beanstalk'. A red arrow points to the 'Elastic Beanstalk' service card, which is highlighted with a blue border. The card displays the text 'Run and Manage Web Apps'. Below the service cards, there are sections for 'Features' and 'Resources'. A modal dialog at the bottom right provides information about cross-Region search for resources.

Services

- Elastic Beanstalk
- Elastic Transcoder
- Elastic Container Service

Features

- Applications
- Environments
- Elastic IPs

Resources in us-east-1 / for a focused search

Were these results helpful?

Yes   No

Looking for resources in other Regions?  
You can enable cross-Region search for resources across all Regions in your account by specifying an aggregator index.

## Step 59: Create a New Application

Click **Create Application** to begin setting up a new Elastic Beanstalk application:

The screenshot shows the Amazon Elastic Beanstalk landing page. At the top right, there is a 'Get started' button with a red arrow pointing to it. Below the main heading, there is a section titled 'Benefits and features' with two columns: 'Easy to get started' and 'Complete resource control'. The bottom of the page includes navigation links like CloudShell, Feedback, and Console Mobile App, along with copyright and privacy information.

## Step 60: Select Web Server Environment

Select **Web Server Environment**, enter your **application name**, and scroll down to configure the environment:

The screenshot shows the 'Configure environment' step of the AWS Elastic Beanstalk 'Create environment' wizard. The left sidebar lists steps: Step 1 (Configure environment, which is selected), Step 2 (Configure service access), Step 3 - optional (Set up networking, database, and tags), Step 4 - optional (Configure instance traffic and scaling), Step 5 - optional (Configure updates, monitoring, and logging), and Step 6 (Review). The main area is titled 'Configure environment' with an 'Info' link. It shows the 'Environment tier' set to 'Web server environment' (selected) and 'Worker environment' (unselected). The 'Application information' section shows the 'Application name' as 'github-actions-demo'. The 'Environment information' section shows the 'Environment name' as 'Github-actions-demo-env' and the 'Domain' as '.us-east-1.elasticbeanstalk.com'. A 'Check availability' button is present. The bottom navigation bar includes CloudShell, Feedback, and Console Mobile App.

Great! Let's continue the README with **steps 61–70**, keeping the same clear and professional style.

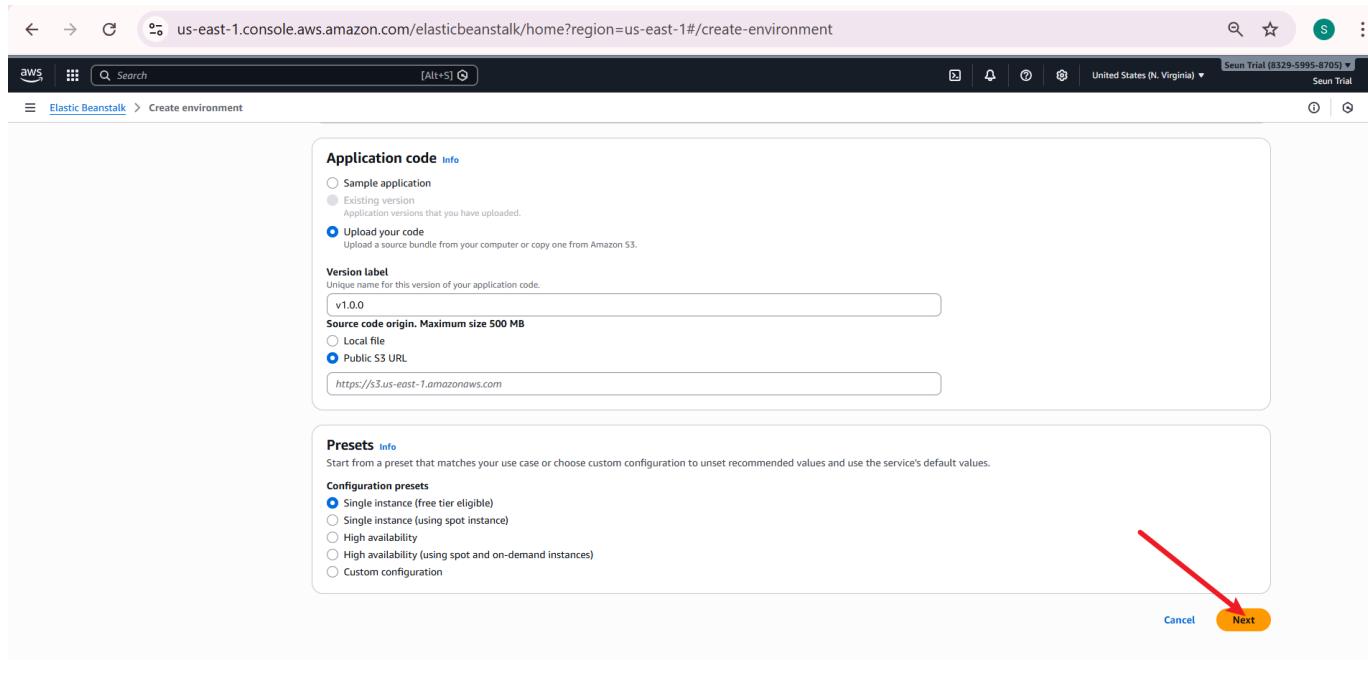
## Step 61: Select Node.js Platform and Upload Code

Choose the **Node.js platform version 20**, select **Upload code from S3**, and add a deployment label (e.g., **v1.0.0**):

The screenshot shows the 'Platform' step of the AWS Elastic Beanstalk 'Create environment' wizard. The 'Platform' dropdown is set to 'Node.js'. The 'Platform branch' dropdown is set to 'Node.js 20 running on 64bit Amazon Linux 2023'. The 'Platform version' dropdown is set to '6.7.2 (Recommended)'. The 'Application code' section shows the 'Upload your code' option selected, with a 'Version label' of 'v1.0.0' and a 'Source code origin' of 'Public S3 URL' with the URL 'https://s3.us-east-1.amazonaws.com'. The bottom navigation bar includes CloudShell, Feedback, and Console Mobile App.

## Step 62: Click Next

Click **Next** to proceed to the next configuration step:



## Step 63: Configure Service Access

In the **Configure service access** step, click **Create Service Role**:

The screenshot shows the AWS Elastic Beanstalk 'Create environment' wizard at Step 2: 'Configure service access'. The left sidebar lists steps 1 through 6. Step 2 is selected, showing the 'Configure service access' page. The main content area is titled 'Service access' and describes IAM roles for Elastic Beanstalk. It includes three sections: 'Service role', 'EC2 instance profile', and 'EC2 key pair - optional'. Each section has a dropdown menu and a 'Create role' button. A red box highlights the 'Choose a service role' dropdown, and a red arrow points to the 'Create role' button in the 'Service role' section.

## Step 64: Set Use Case to Elastic Beanstalk

Ensure the **use case** is set to **Elastic Beanstalk**, then click **Next**:

**Select trusted entity**

**Trusted entity type**

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

Elastic Beanstalk

Choose a use case for the specified service.  
Use case

- Elastic Beanstalk - Compute Allows your environment's EC2 instances to perform operations required for your application.
- Elastic Beanstalk - Environment Allows access to other AWS service resources that are required to create and manage environments.

Cancel Next

## Step 65: Create Service Role with Appropriate Permissions

Review the permissions and click **Next** → **Review** → **Create** to create the service role:

**Add permissions**

**Permissions policies (2)**

The type of role that you selected requires the following policy.

Policy name	Type
AWSelasticBeanstalkEnhancedHealth	AWS managed
AWSelasticBeanstalkManagedUpdatesCustomerRolePolicy	AWS managed

▶ Set permissions boundary - optional

Cancel Previous Next

## Step 66: Switch Back to Elastic Beanstalk Creation

Once the role is created, switch back to the **Elastic Beanstalk application creation** tab:

The screenshot shows the AWS IAM Roles page. A red arrow points to the browser tab title 'Configure service access | Elastic...'. The main content area displays a table of roles, with the first row, 'aws-elasticbeanstalk-service-role', highlighted. The table columns are 'Role name', 'Trusted entities', and 'Last activity'. Below the table, there are sections for 'Roles Anywhere' and 'Access AWS from your non AWS workloads'.

Role name	Trusted entities	Last activity
<a href="#">aws-elasticbeanstalk-service-role</a>	AWS Service: elasticbeanstalk	-
<a href="#">AWSServiceRoleForAmazonElasticFileSystem</a>	AWS Service: elasticfilesystem (Service-Linked Role)	36 days ago
<a href="#">AWSServiceRoleForAutoScaling</a>	AWS Service: autoscaling (Service-Linked Role)	36 days ago
<a href="#">AWSServiceRoleForElasticLoadBalancing</a>	AWS Service: elasticloadbalancing (Service-Linked Role)	36 days ago
<a href="#">AWSServiceRoleForRDS</a>	AWS Service: rds (Service-Linked Role)	28 minutes ago
<a href="#">AWSServiceRoleForResourceExplorer</a>	AWS Service: resource-explorer-2 (Service-Linked Role)	31 minutes ago
<a href="#">AWSServiceRoleForSupport</a>	AWS Service: support (Service-Linked Role)	-
<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS Service: trustedadvisor (Service-Linked Role)	-
<a href="#">EC2SSMRole</a>	AWS Service: ec2	-

## Step 67: Create EC2 Instance Profile Role

Select **aws-elasticbeanstalk-service-role** and click to **create an EC2 instance profile role**:

The screenshot shows the 'Configure service access' step of the Elastic Beanstalk environment creation wizard. A red arrow points to the 'Create role' button under the 'Service role' section. The 'EC2 instance profile' and 'EC2 key pair - optional' sections also have 'Create role' buttons. At the bottom right, there are 'Cancel', 'Skip to review', 'Previous', and 'Next' buttons.

## Step 68: Ensure Elastic Beanstalk Compute is Selected

Ensure **Elastic Beanstalk compute** is selected, then click **Next**:

Step 1  
 **Select trusted entity** Info  
 Step 2  
 Add permissions  
 Step 3  
 Name, review, and create

**Select trusted entity** Info

**Trusted entity type**

- AWS service** Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account** Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity** Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

- SAML 2.0 federation** Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy** Create a custom trust policy to enable others to perform actions in this account.

**Use case**

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

Elastic Beanstalk

**Choose a use case for the specified service.**

**Use case**

- Elastic Beanstalk - Compute** Allows your environment's EC2 instances to perform operations required for your application.
- Elastic Beanstalk - Environment** Allows access to other AWS service resources that are required to create and manage environments.

**Cancel** **Next**

## Step 68b: Review EC2 Instance Profile Role

Ensure the appropriate permissions are selected, review, and click **Create Role**:

Step 1  
 **Select trusted entity**  
 Step 2  
 **Add permissions**  
 Step 3  
 Name, review, and create

**Add permissions** Info

**Permissions policies (3)** Info

The type of role that you selected requires the following policy.

Policy name	Type
<input checked="" type="checkbox"/> AWSElasticBeanstalkMulticontainerDocker	AWS managed
<input checked="" type="checkbox"/> AWSElasticBeanstalkWebTier	AWS managed
<input checked="" type="checkbox"/> AWSElasticBeanstalkWorkerTier	AWS managed

**Set permissions boundary - optional**

**Cancel** **Previous** **Next**

## Step 69: Assign Roles and Key Pair

Select the created **service role**, click **Refresh** if the role is missing, then select the **EC2 key pair**, and click **Next**:

**Configure service access**

**Service access**

IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

**Service role**

Choose an IAM role for Elastic Beanstalk to assume as a service role. The IAM role must have the required IAM managed policies.

aws-elasticbeanstalk-service-role

**Create role**

**EC2 instance profile**

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

aws-elasticbeanstalk-ec2-role

**Create role**

**EC2 key pair - optional**

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

MyKeyPair

**Next**

## Step 70: Configure VPC and Subnets

Select the **default VPC** and its subnets, and enable **public IP**:

**Instance settings**

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. [Learn more](#)

**VPC**

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. [Learn more](#)

vpc-0abae1d7e8bb3d94 | (172.31.0.0/16)

**Create VPC**

**Public IP address**

Assign a public IP address to the Amazon EC2 instances in your environment.

Enable

**Instance subnets**

Availability Zone	Subnet	CIDR	Name
Deselect all instance subnets	subnet-06bf8e21163d81474	172.31.16.0/20	
us-east-2c	subnet-0775b84bcc32b6507	172.31.32.0/20	
us-east-2a	subnet-0a0325c62371a1a20	172.31.0.0/20	

**Database**

Integrate an RDS SQL database with your environment. [Learn more](#)

Enable database

## Step 71: Add Project and Environment Tags

Add **Project** and **Environment** tags for your Elastic Beanstalk environment, then click **Next**:

The screenshot shows the 'Create environment' step in the AWS Elastic Beanstalk console. In the top section, three subnets are selected: us-east-2b, us-east-2c, and us-east-2a, each associated with a specific IP range and subnet ID. Below this, there's a 'Database' section with an 'Info' link and a toggle switch for enabling the database. Under 'Tags', two tags are defined: 'Project' with value 'nodejs-ci-cd' and 'Environment' with value 'dev'. At the bottom right, there are 'Cancel', 'Skip to review', 'Previous', and a highlighted 'Next' button.

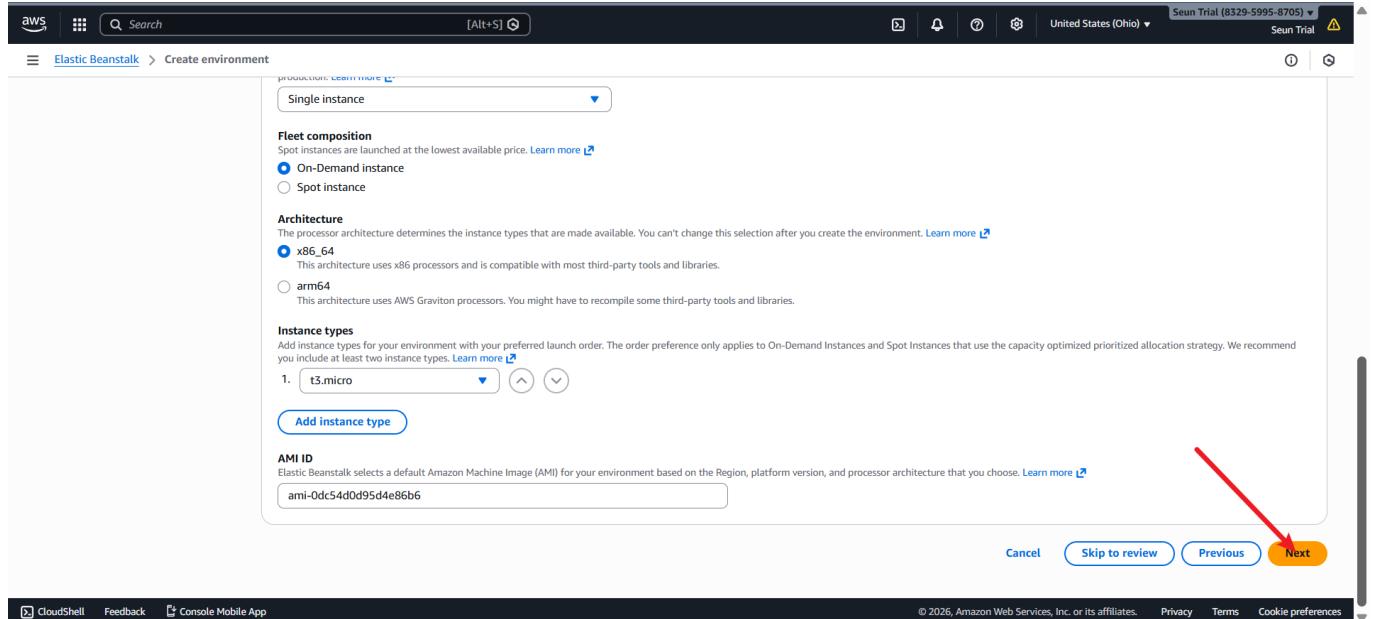
## Step 72: Configure Root Volume and EC2 Settings

Leave the **root volume** at default, enable **only IMDSv1**, set CloudWatch monitoring interval to **5 minutes**, and select the **default EC2 security group**:

The screenshot shows the 'Configure instance traffic and scaling' step. On the left, a sidebar indicates 'Step 5 - optional' and 'Configure updates, monitoring, and logging'. The main area shows settings for instance traffic: 'each instance' (Container default), 'to each instance' (100 IOPS), and 'volume attached to your environment's EC2 instance' (125 MiB/s). Under 'Amazon CloudWatch monitoring', the 'Monitoring interval' is set to '5 minute'. In the 'IMDSv1' section, it says 'With the current setting, the environment enables only IMDSv2.' and has a checked checkbox for 'Disable'. The 'EC2 security groups' section shows a dropdown menu with 'Choose security groups' and a selected item 'sg-0e3ece17a71b13d4b | default'.

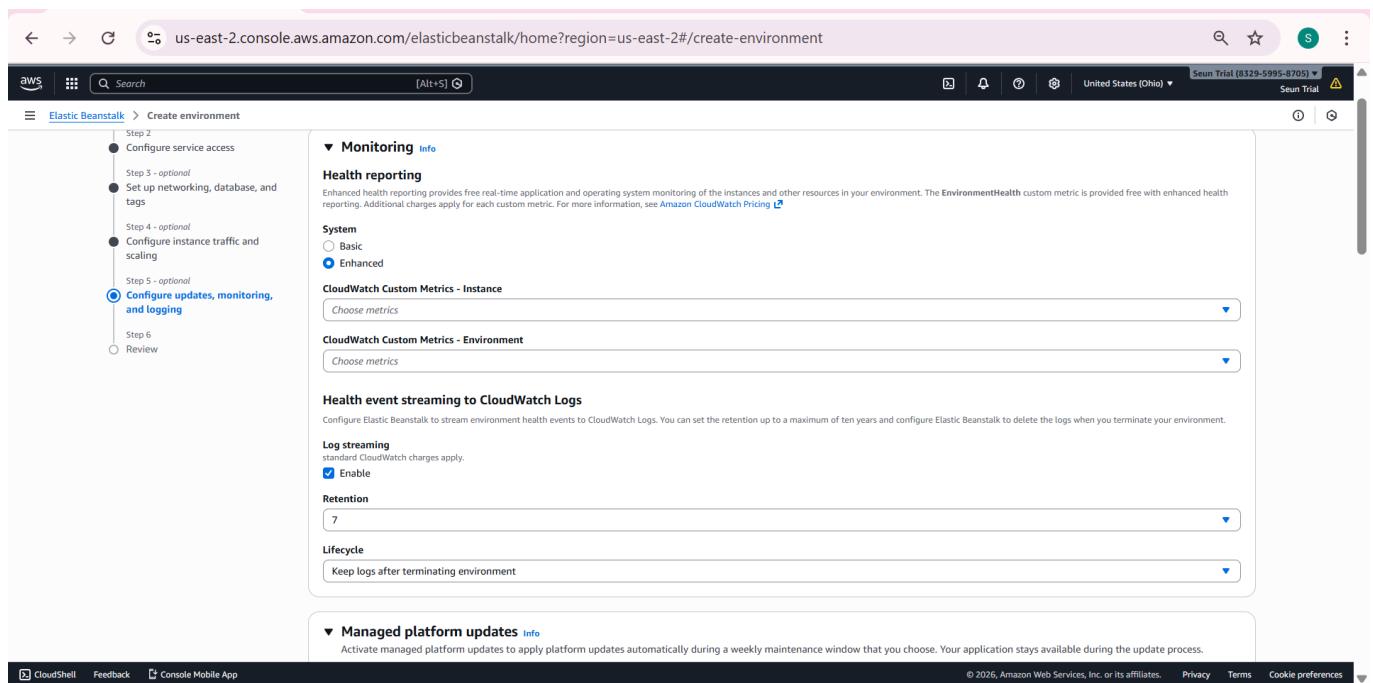
## Step 73: Set Instance Type and Scaling

Choose **Single instance, On-Demand**, architecture **x86\_64**, instance type **t3.micro**, and click **Next**:



## Step 74: Configure Monitoring

Set **enhanced health monitoring**, CloudWatch metrics, and other monitoring options according to your requirements:



## Step 75: Enable Managed Platform Updates

Enable **managed platform updates**, select the weekly maintenance window, and choose **minor and patch updates**:

**Managed platform updates**

- Activate managed platform updates to apply platform updates automatically during a weekly maintenance window that you choose. Your application stays available during the update process.
- Managed updates**:  Enable
- Weekly update window**: Thursday at 01:04 UTC
- Update level**: Minor and patch
- Instance replacement**: If enabled, an instance replacement will be scheduled if no other updates are available.  Enable

**Email notifications**

Enter an email address to receive email notifications for important events from your environment. [Learn more](#)

**Email**: old\_email@mail.com

**Rolling updates and deployments**

Choose how Amazon Elastic Beanstalk propagates source code changes and software configuration updates. [Learn more](#)

**Application deployments**

Choose how Amazon Elastic Beanstalk propagates source code changes and software configuration updates. [Learn more](#)

**Deployment policy**: All at once

**Batch size type**: Percentage

## Step 75b: Configure Rolling Updates and Deployment

Set rolling update policies and deployment preferences to control how instances are updated:

**Rolling updates and deployments**

Choose how Amazon Elastic Beanstalk propagates source code changes and software configuration updates. [Learn more](#)

**Application deployments**

Choose how Amazon Elastic Beanstalk propagates source code changes and software configuration updates. [Learn more](#)

**Deployment policy**: All at once

**Batch size type**: Percentage

**Deployment batch size**: 100 % instances at a time

**Configuration updates**

Changes to virtual machine settings and VPC configuration trigger rolling updates to replace the instances in your environment without downtime. [Learn more](#)

**Rolling update type**: Disabled

**Deployment preferences**

Customize health check requirements and deployment timeouts.

**Ignore health check**: Don't fail deployments due to health check failures.  False

**Health threshold**: Lower the threshold for an instance in a batch to pass health checks during an update or deployment.  Ok

## Step 76: Configure Platform Software Options

Set platform-specific options such as **proxy server**, **X-Ray**, **S3 log storage**, and **instance log streaming**:

The screenshot shows the 'Platform software' section of the AWS Elastic Beanstalk configuration interface. It includes options for 'Proxy server' (set to Nginx), 'Amazon X-Ray' (disabled), 'X-Ray daemon' (disabled), 'S3 log storage' (disabled), 'Rotate logs' (disabled), 'Log streaming' (disabled), and 'Environment properties' (empty). The 'Next Step' button at the bottom is highlighted.

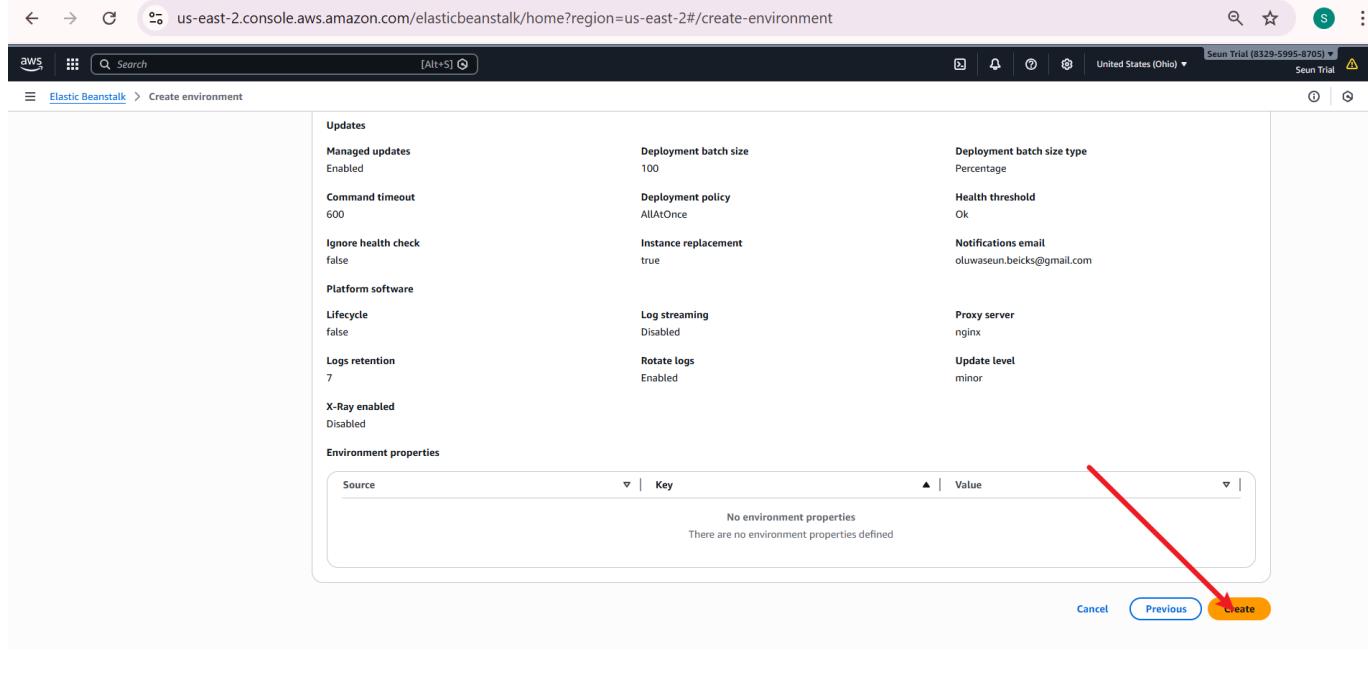
## Step 77: Click Next

After reviewing optional settings, click **Next** to proceed to the final step:

The screenshot shows the same configuration page as above, but with a red arrow pointing to the 'Next' button at the bottom right. The 'Environment properties' section is also highlighted.

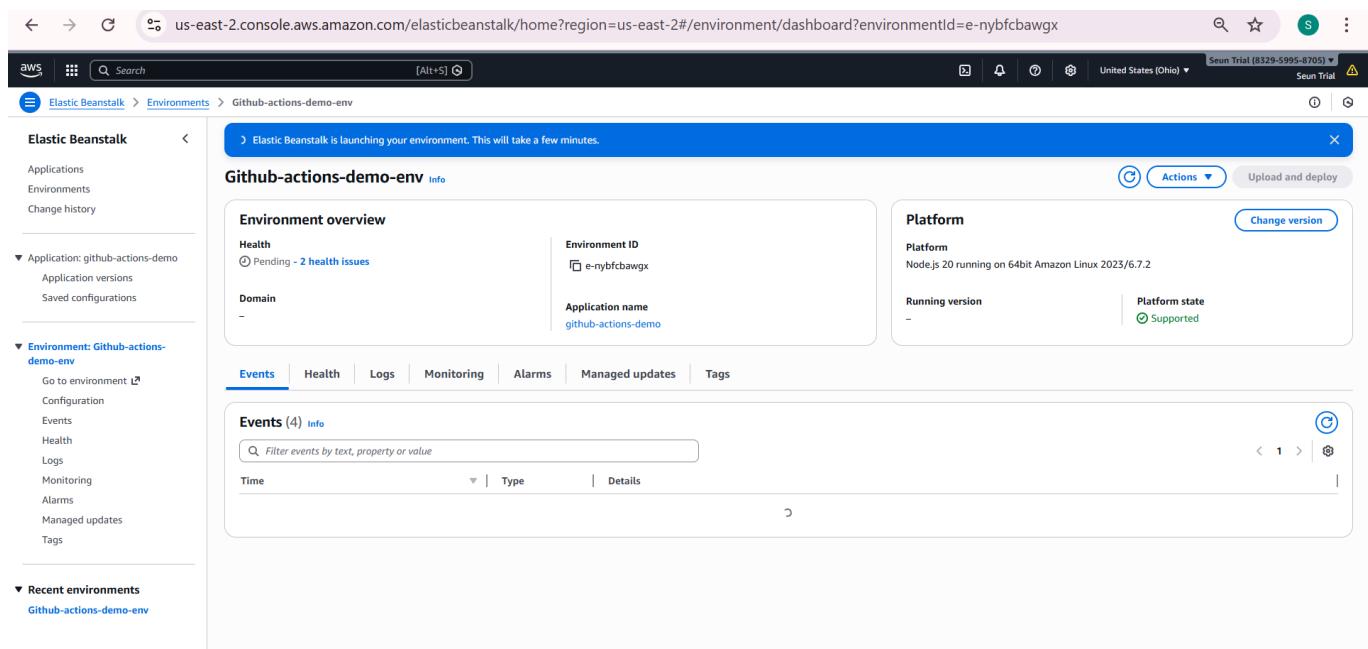
## Step 78: Review Configuration and Create Environment

Review all settings, verify that everything is correct, and click **Create**:



## Step 79: Environment Creation in Progress

The Elastic Beanstalk environment is being created:



## Step 80: Environment Successfully Launched

Once the environment launches successfully, copy the provided **Elastic Beanstalk domain** and open it in a new browser tab to verify deployment:

The screenshot shows the AWS Elastic Beanstalk console with the environment 'Github-actions-demo-env' successfully launched. The domain listed is `github-actions-demo-env.eba-h8vivgm.us-east-2.elasticbeanstalk.com`. The 'Events' tab is selected, showing 12 events.

## Step 81: Welcome to Elastic Beanstalk Page

Visit your Elastic Beanstalk environment's domain to verify the application is running successfully:

The screenshot shows the AWS Elastic Beanstalk application welcome page. The title is 'Welcome to Your Elastic Beanstalk Application'. The message says: 'Congratulations! Your Node.js application is now running on your own dedicated environment in the AWS Cloud.' There is a 'Learn More' button.

## Benefits of AWS Elastic Beanstalk

Discover why thousands of developers rely on AWS Elastic Beanstalk to deploy and manage their

## Step 82: Add Deployment Workflow

Create a **deployment workflow** in `.github/workflows` for deploying the application to Elastic Beanstalk:

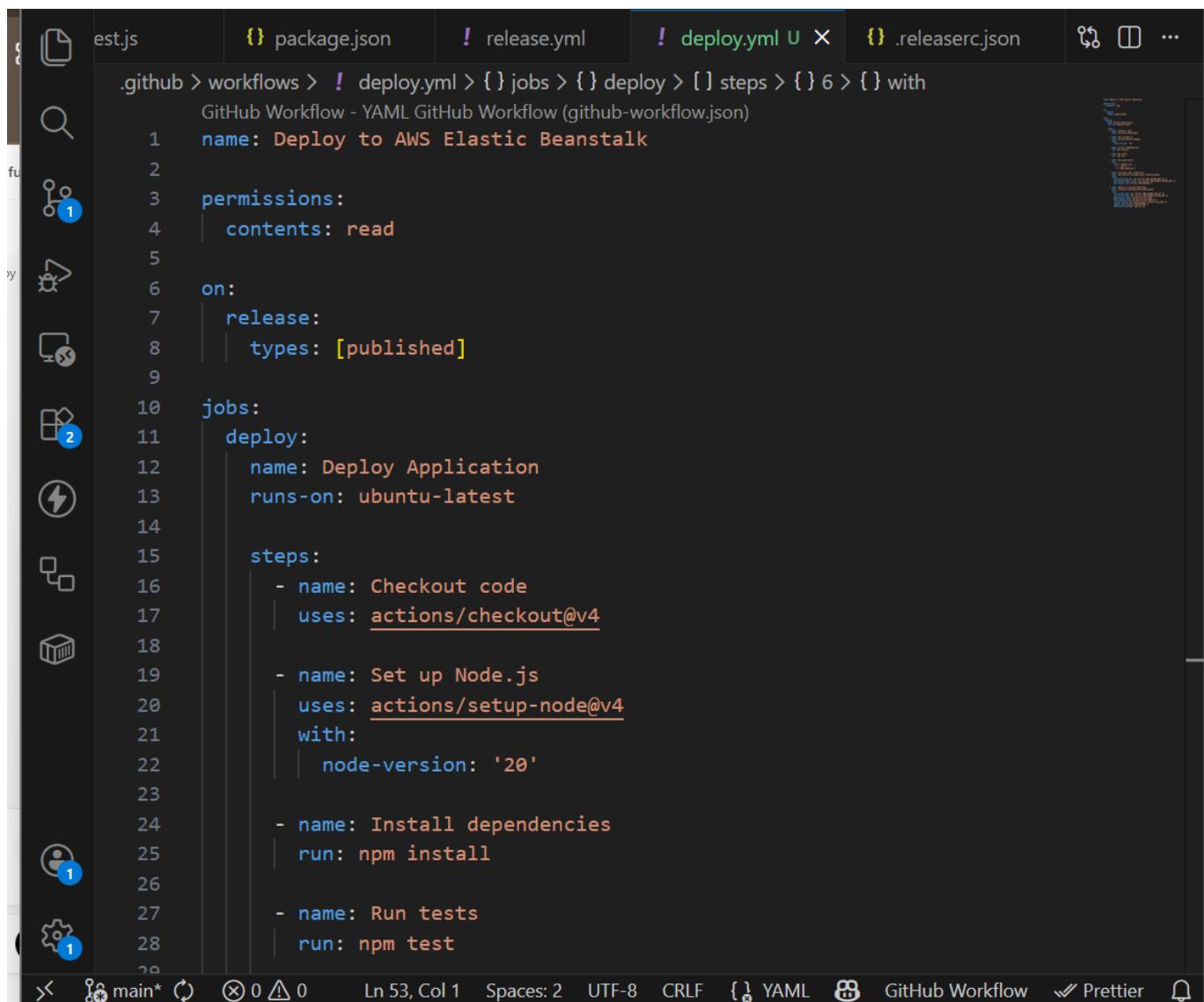
```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ touch .github/workflows/deploy.yml
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 83: Configure `deploy.yml` File

Edit the `deploy.yml` workflow file with the steps to **checkout code, install dependencies, run tests, zip the application, configure AWS credentials, and deploy to Elastic Beanstalk**:



```
.github > workflows > ! deploy.yml > {} jobs > {} deploy > [] steps > {} 6 > {} with
  GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
    1   name: Deploy to AWS Elastic Beanstalk
    2
    3   permissions:
    4     contents: read
    5
    6   on:
    7     release:
    8       types: [published]
    9
   10  jobs:
   11    deploy:
   12      name: Deploy Application
   13      runs-on: ubuntu-latest
   14
   15    steps:
   16      - name: Checkout code
   17        uses: actions/checkout@v4
   18
   19      - name: Set up Node.js
   20        uses: actions/setup-node@v4
   21        with:
   22          node-version: '20'
   23
   24      - name: Install dependencies
   25        run: npm install
   26
   27      - name: Run tests
   28        run: npm test
```

## Step 84: Commit and Push Deployment Workflow

After creating the deployment workflow file, **commit and push** it to your repository:

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo □ >

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .github/workflows/deploy.yml

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "feat(deploy): add aws elastic beanstalk deployment pipeline"
[main 593af8a] feat(deploy): add aws elastic beanstalk deployment pipeline
 1 file changed, 52 insertions(+)
 create mode 100644 .github/workflows/deploy.yml

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 990 bytes | 247.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Oluwaseunoa/github-actions-demo.git
 87697ee..593af8a main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 85: Edit `app.js` to Trigger Deployment Workflow

Make a small **change in `app.js`** to trigger the deployment workflow when pushed:

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

app.get('/', (req, res) => {
  res.send('Hello, World! Deployed automatically to AWS 🎉');
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

## Step 86: Commit and Push Changes

Commit and push the changes to the repository to trigger the deployment workflow:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add app.js

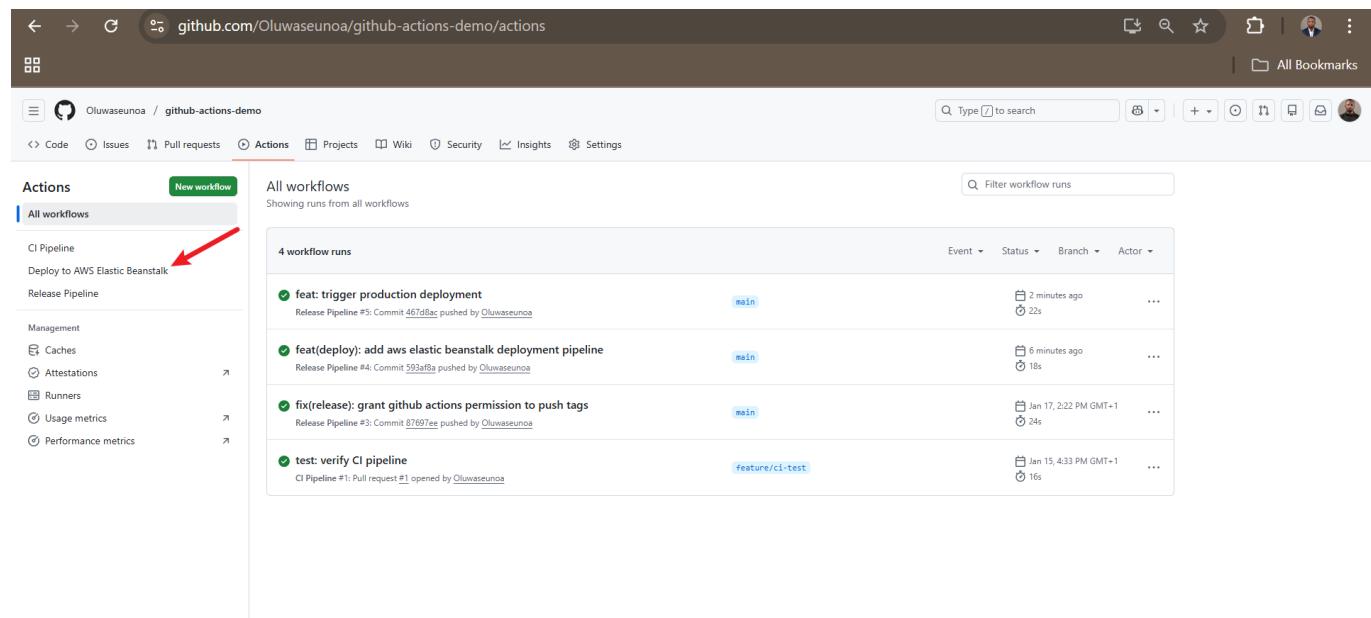
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "feat: trigger production deployment"
[main 467d8ac] feat: trigger production deployment
 1 file changed, 1 insertion(+), 1 deletion(-)

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 172.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Oluwaseunoa/github-actions-demo.git
 593af8a..467d8ac  main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 87: Navigate to Repository Actions

Go to the **Actions tab** in your GitHub repository and click on the **Deploy to Elastic Beanstalk workflow**:

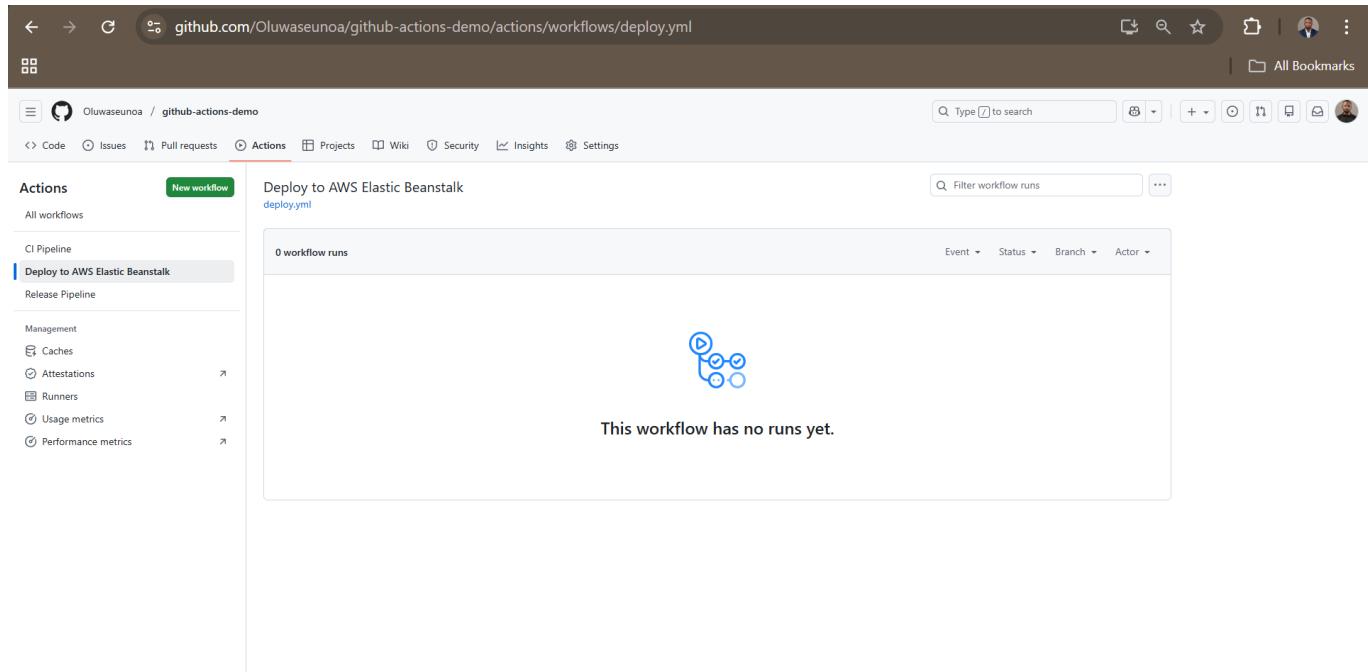


The screenshot shows the GitHub Actions tab for the repository 'github-actions-demo'. On the left sidebar, under 'Actions', there is a list of pipelines: 'CI Pipeline', 'Deploy to AWS Elastic Beanstalk' (which has a red arrow pointing to it), 'Release Pipeline', 'Management' (with sub-options 'Caches', 'Attestations', 'Runners', 'Usage metrics', and 'Performance metrics'). The main area displays the 'All workflows' section, which lists four workflow runs:

Workflow	Branch	Event	Status	Actor
feat: trigger production deployment	main	2 minutes ago	<span>22s</span>	...
feat(deploy): add aws elastic beanstalk deployment pipeline	main	6 minutes ago	<span>18s</span>	...
fix(release): grant github actions permission to push tags	main	Jan 17, 2:22 PM GMT+1	<span>24s</span>	...
test: verify CI pipeline	feature/ci-test	Jan 15, 4:33 PM GMT+1	<span>16s</span>	...

## Step 88: Workflow Has Not Run Yet

Since the workflow is newly configured, it will show "**This workflow has no runs yet**":



The screenshot shows a GitHub Actions workflow named "Deploy to AWS Elastic Beanstalk" in the repository "Oluwaseunoa/github-actions-demo". The workflow file is "deploy.yml". The interface displays 0 workflow runs, with a message stating "This workflow has no runs yet." A blue icon of a person carrying a briefcase is centered below the message. The left sidebar shows navigation options like "Actions", "All workflows", "CI Pipeline", "Deploy to AWS Elastic Beanstalk", "Release Pipeline", and "Management".

## Step 89: Update `deploy.yml` with Write Permission

Edit the `deploy.yml` to ensure the workflow has **write permission to push deployment tags or trigger further actions**:

```
.github > workflows > ! deploy.yml > {} permissions > abc contents
GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
1 name: Deploy to AWS Elastic Beanstalk
2
3 permissions:
4   contents: write
5
6 on:
7   release:
8     types: [published]
9
10 jobs:
11   deploy:
12     name: Deploy Application
13     runs-on: ubuntu-latest
14
15 steps:
16   - name: Checkout code
17     uses: actions/checkout@v4
18
19   - name: Set up Node.js
20     uses: actions/setup-node@v4
21     with:
22       node-version: '20'
23
24   - name: Install dependencies
25     run: npm install
26
27   - name: Run tests
28     run: npm test
29
```

---

## Step 90: Push and Commit Changes

Push and commit the changes to GitHub to activate the workflow:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .github/workflows/deploy.yml

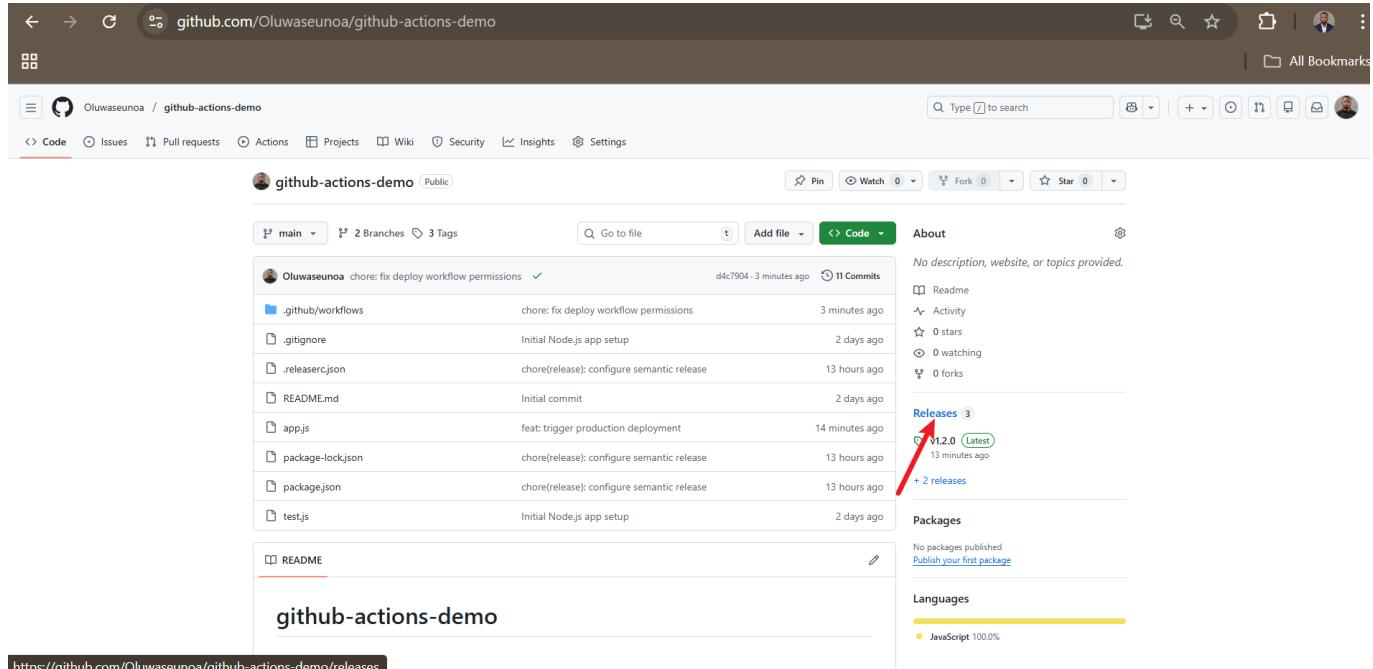
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "chore: fix deploy workflow permissions"
[main d4c7904] chore: fix deploy workflow permissions
 1 file changed, 1 insertion(+), 1 deletion(-)

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 487 bytes | 162.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Oluwaseunoa/github-actions-demo.git
  467d8ac..d4c7904 main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

## Step 91: Navigate to Repository Releases

Go to your GitHub repository **Releases tab** to view existing releases:



The screenshot shows the GitHub repository page for 'github-actions-demo'. The URL in the address bar is 'https://github.com/Oluwaseunoa/github-actions-demo'. The page navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area displays the repository's structure with branches (main, 2 Branches) and tags (3 Tags). A list of recent commits is shown, each with a timestamp and author. On the right side, there are sections for About (no description), Activity (0 stars, 0 watching, 0 forks), Releases (3 releases, including v1.2.0 (Latest) created 13 minutes ago), Packages (no packages published), and Languages (JavaScript 100.0%). A red arrow points to the 'v1.2.0 (Latest)' release link.

## Step 92: New Release Appears

The newly created release is visible, showing the **version number and commit**:

The screenshot shows the GitHub Releases page for the repository `github-actions-demo`. It displays two releases:

- v1.2.0 (Latest)**: Published 15 minutes ago. Includes a note about triggering a production deployment and two source code assets (zip and tar.gz).
- v1.1.0**: Published 19 minutes ago.

## Step 93: Workflow Did Not Run

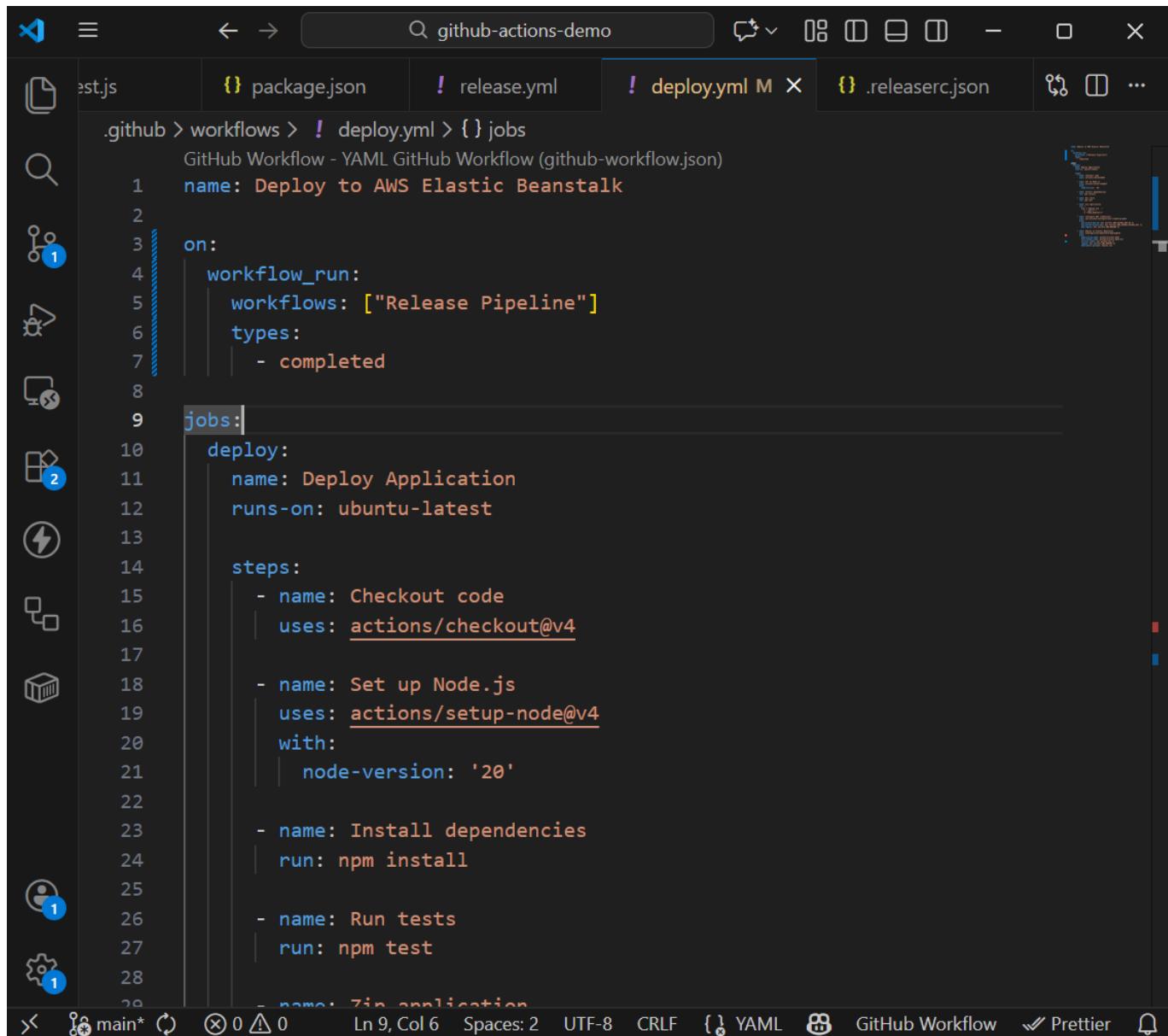
At this point, the deployment workflow did **not run automatically**, indicating a configuration mismatch:

The screenshot shows the GitHub Actions Pipelines page for the repository `github-actions-demo`. The sidebar shows the **Actions** section with the **Deploy to AWS Elastic Beanstalk** workflow selected. The main area shows the workflow configuration and a summary of runs:

- Deploy to AWS Elastic Beanstalk**: `deploy.yml`
- 0 workflow runs**: A message states "This workflow has no runs yet."

## Step 94: Update `deploy.yml` for Release Pipeline

Edit the `deploy.yml` workflow to **match the release pipeline name** and ensure **unique version labels** for Elastic Beanstalk deployments:



```
.github > workflows > ! deploy.yml > {} jobs
  GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
  1   name: Deploy to AWS Elastic Beanstalk
  2
  3   on:
  4     workflow_run:
  5       workflows: ["Release Pipeline"]
  6       types:
  7         - completed
  8
  9   jobs:
 10     deploy:
 11       name: Deploy Application
 12       runs-on: ubuntu-latest
 13
 14       steps:
 15         - name: Checkout code
 16           uses: actions/checkout@v4
 17
 18         - name: Set up Node.js
 19           uses: actions/setup-node@v4
 20           with:
 21             node-version: '20'
 22
 23         - name: Install dependencies
 24           run: npm install
 25
 26         - name: Run tests
 27           run: npm test
 28
 29         - name: Zip application
 30           uses: actions/zip@v1
```

---

## Step 95: Commit and Push Changes

Commit and push the updated workflow to GitHub:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git add .github/workflows/deploy.yml

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit -m "fix: trigger deployment after release pipeline"
[main 0b383a1] fix: trigger deployment after release pipeline
 1 file changed, 5 insertions(+), 8 deletions(-)

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 567 bytes | 189.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Oluwaseunoa/github-actions-demo.git
  d4c7904..0b383a1  main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 96: Make a New Feature Commit

Create a **new feature commit** to trigger the deployment pipeline and push it to the repository:

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git commit --allow-empty -m "feat: trigger production deployment"
[main fcebe97] feat: trigger production deployment

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$ git push
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 215 bytes | 215.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Oluwaseunoa/github-actions-demo.git
  0b383a1..fcebe97  main -> main

HP@DESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/GitHub-Actions-Projects/github-actions-demo (main)
$
```

---

## Step 97: Release Version v1.3.0

Create a new release on GitHub to version the application as **v1.3.0**:

The screenshot shows the GitHub repository page for 'github-actions-demo'. The 'Actions' tab is active, displaying a list of workflow runs. One specific workflow run, 'Deploy to AWS Elastic Beanstalk', is highlighted with a red box. The run was completed by Oluwaseunoa and took 52 seconds. Other workflow runs listed include 'deploy to ready environment', 'Deploy to AWS Elastic Beanstalk', 'feat: deploy production build', 'Deploy to AWS Elastic Beanstalk', 'fix: explicitly pass aws credentials to beanstalk deploy action', and 'Deploy to AWS Elastic Beanstalk'.

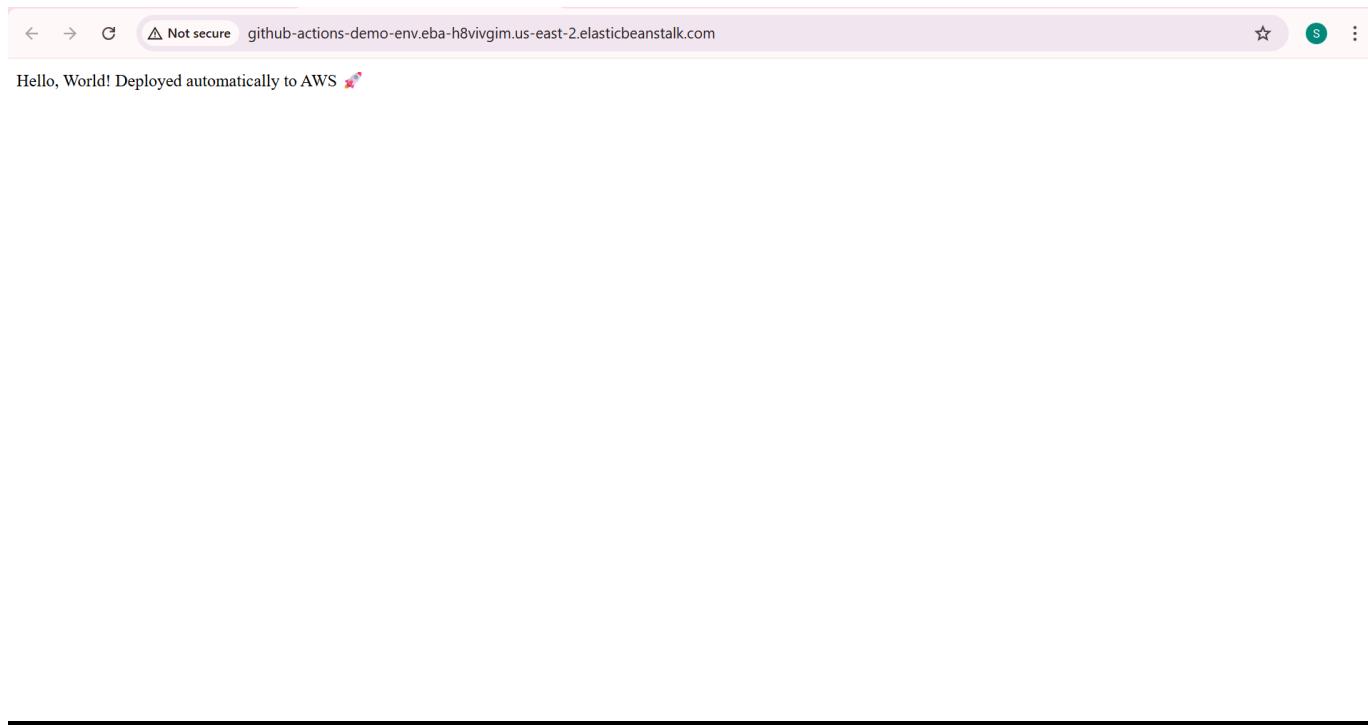
## Step 98: Deploy to AWS Elastic Beanstalk Pipeline Runs Successfully

The **Deploy to EBS workflow** now runs successfully, deploying the latest application version:

The screenshot shows the GitHub Actions page for the 'github-actions-demo' repository. The 'All workflows' section is selected, showing a list of 17 workflow runs. One workflow run, 'Deploy to AWS Elastic Beanstalk', is highlighted with a red box. The run was completed by Oluwaseunoa and took 52 seconds. Other workflow runs listed include 'deploy to ready environment', 'Deploy to AWS Elastic Beanstalk', 'feat: deploy production build', 'Deploy to AWS Elastic Beanstalk', 'fix: explicitly pass aws credentials to beanstalk deploy action', and 'Deploy to AWS Elastic Beanstalk'.

## Step 99: Application Live on Elastic Beanstalk

Finally, visit the **Elastic Beanstalk URL** to confirm the application is **live and accessible**:



---

## CI/CD Workflow Summary

1. **CI pipeline:** Runs tests on every PR or push.
  2. **Semantic release pipeline:** Automatically generates release notes and versions using conventional commits.
  3. **Deploy to Elastic Beanstalk:** Triggered on release, builds zip, uploads to S3, deploys to EBS.
- 

## References

- [GitHub Actions Documentation](#)
- [AWS Elastic Beanstalk Documentation](#)
- [Semantic Release](#)
- [einaregilsson/beanstalk-deploy](#)