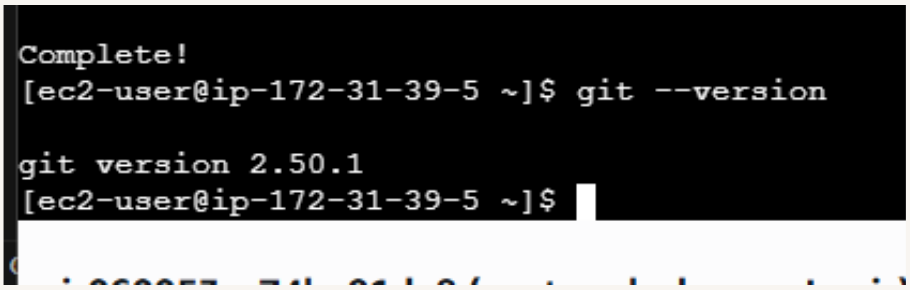
Connect a GitHub Repo with AWS

Louis Moyo



Louis Moyo
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to set up Git and GitHub, connect my web application to a GitHub repository, and manage my code changes through version control. I'm doing this project to learn how developers collaborate on shared codebases, track every change made to an application, and build the foundation for a CI/CD pipeline that can automatically build and deploy updates. This will help me understand the importance of repositories in DevOps workflows and give me hands-on practice using Git commands and GitHub for real-world projects.

Key tools and concepts

Services I used were AWS EC2 for hosting, GitHub for version control, and VS Code Remote SSH to work directly on my EC2 instance. Key concepts I learnt include how to: Create and push commits from a remote environment into GitHub. Resolve issues like large file pushes, branch mismatches, and submodules. Write a README.md in Markdown with proper formatting and links. Exclude unnecessary files from Git using .gitignore. Understand the basics of a CI/CD pipeline foundation.

Project reflection

This project took me approximately 2 hours. The most challenging part was fixing Git errors (like pushing large VS Code server files or handling branch conflicts). It was most rewarding to see my changes appear in my GitHub repository and to end up with a professional README file that showcases my work clearly.

L

Louis Moyo
NextWork Student

I did this project because I wanted to practice hands-on DevOps fundamentals and build confidence in using Git, GitHub, and AWS together. It also aligns with my long-term goal of becoming a cloud security and automation consultant, so every step I take helps me gain the skills I'll need with real clients.

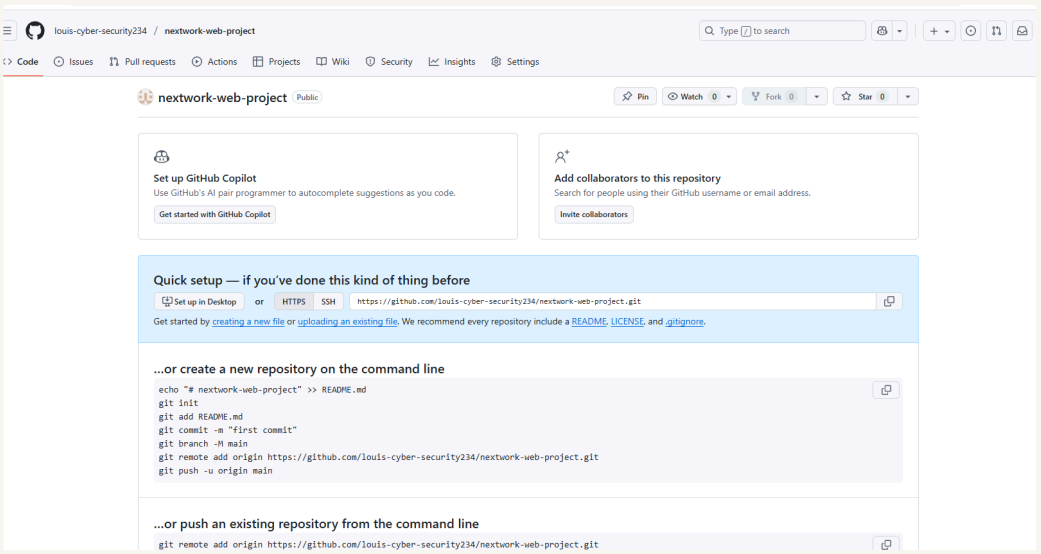
This project is part two of a series of DevOps projects where I'm building a complete CI/CD pipeline. I'll be working on the next project tomorrow, which builds on what I've already done here-moving from code storage and documentation into actual automation of builds, tests, and deployments using AWS tools.

L
Louis Moyo
NextWork Student

Git and GitHub

Git is a version control system that tracks changes in code, like a time machine for developers. It helps manage versions, roll back if needed, and collaborate with teams. I installed Git on my EC2 instance by updating packages and running: sudo apt update sudo apt install git -y git --version This confirmed Git was installed and ready for use with my project.

GitHub is a cloud-based platform that stores Git repositories, making it easy to manage versions of code, collaborate with others, and track changes visually. It acts as a central hub where developers can push, pull, and review code together. I'm using GitHub in this project to securely store my web app's code, connect it to my EC2 instance, and build the foundation for a CI/CD pipeline that automates deployment and collaboration.

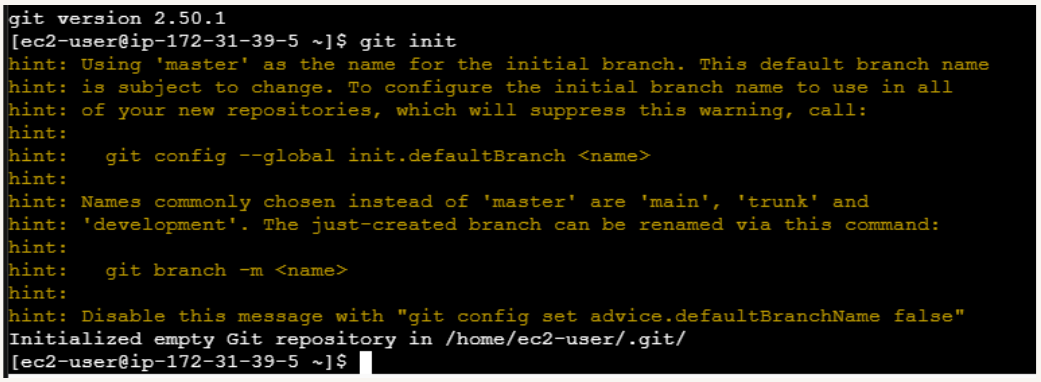

louis-cyber-security234 / nextwork-web-project
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
nextwork-web-project
Public
Pin
Watch 0
Fork 0
Star 0
Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/louis-cyber-security234/nextwork-web-project.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# nextwork-web-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/louis-cyber-security234/nextwork-web-project.git
git push -u origin main
...or push an existing repository from the command line
git remote add origin https://github.com/louis-cyber-security234/nextwork-web-project.git

My local repository

A Git repository is a storage space that keeps all the files for your project along with a full history of changes made to them. It tracks every modification through snapshots called commits, so you can roll back to earlier versions, branch off to test new ideas, and merge updates. Repositories can be local (on your computer or server) or remote (like on GitHub) to make collaboration and version control easy.

git init is a command that creates a new local Git repository in the current folder. It sets up all the hidden configuration files Git needs to start tracking changes to your project. I ran git init in my web app folder on the EC2 instance so Git can begin version-controlling my code.

A branch in Git is like a separate timeline of your project's history where you can make changes without affecting the main codebase. The default branch is usually called main (or sometimes master). After running git init, the terminal response showed that an empty Git repository was created, with the default branch set to main.


git version 2.50.1
[ec2-user@ip-172-31-39-5 ~]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint: git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
Initialized empty Git repository in /home/ec2-user/.git/
[ec2-user@ip-172-31-39-5 ~]$

L
Louis Moyo
NextWork Student

To push local changes to GitHub, I ran three commands

git add

The first command I ran was git add . which stages all the files in my project so they're ready to be included in the next commit. A staging area is like a preparation zone where changes are collected and reviewed before being permanently saved to the project history.

git commit

The second command I ran was git commit -m "Updated index.jsp with new content". This saves the staged changes as a new version (commit) in my project's history. Using -m means I can add a descriptive message about what was changed, which makes it easier to understand the purpose of that commit later.

git push

The third command I ran was git push -u origin master. This uploads (pushes) my committed changes to the remote GitHub repository I named origin. The word master tells Git to push the changes into the master branch. Using -u means I've set the upstream, so in the future I can simply type git push without needing to re-specify the branch or remote.

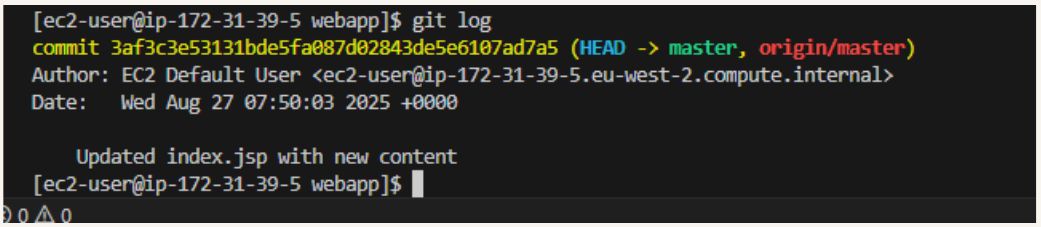
Authentication

When I commit changes to GitHub, Git asks for my credentials because I'm trying to push code from my local repository to a remote repository on GitHub. GitHub requires authentication to make sure that only authorised users can update or change a project.

Local Git identity

Git needs my name and email because every commit must have an author attached to it. This information identifies who made the change and when, which is crucial for collaboration, accountability, and version tracking. It also ensures that when commits are pushed to GitHub, they're correctly linked to my GitHub profile.

Running git log showed me a detailed history of my commits, including the commit hash (unique ID), author name and email, date, and the commit message I wrote. This allows me to review the timeline of changes made to my project and see exactly what was updated in each version.


[ec2-user@ip-172-31-39-5 webapp]$ git log
commit 3af3c3e53131bde5fa087d02843de5e6107ad7a5 (HEAD -> master, origin/master)
Author: EC2 Default User <ec2-user@ip-172-31-39-5.eu-west-2.compute.internal>
Date: Wed Aug 27 07:50:03 2025 +0000
Updated index.jsp with new content
[ec2-user@ip-172-31-39-5 webapp]$
0 0 0

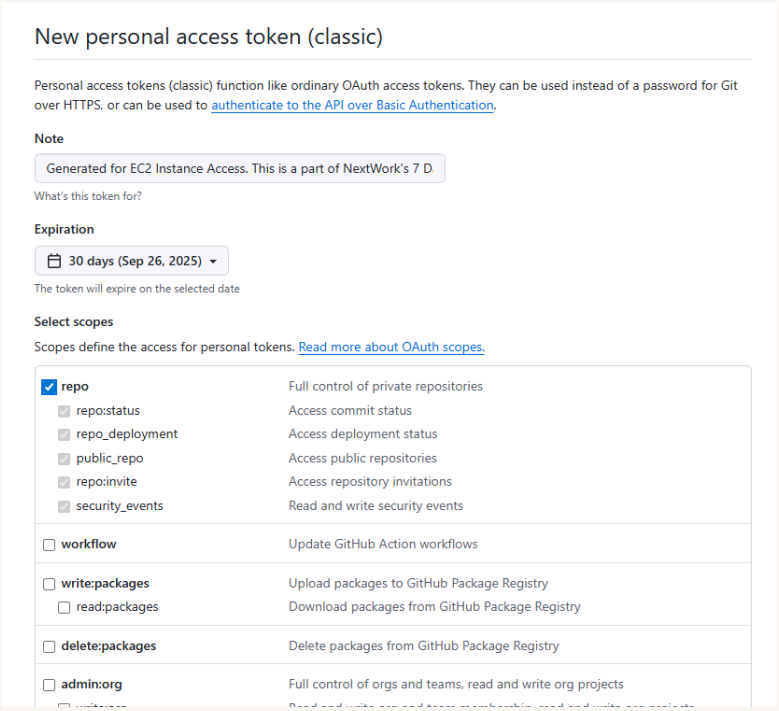
Louis Moyo
NextWork Student

GitHub tokens

GitHub authentication failed when I entered my password because GitHub no longer accepts account passwords for Git operations over HTTPS. Instead, it requires a Personal Access Token (PAT), which acts like a password but is more secure. I need to generate a PAT from my GitHub account settings and use that instead of my normal password when pushing code.

A GitHub token is a secure string of characters (like a digital key) that authorises Git operations without needing to use my GitHub password. It works as a Personal Access Token (PAT) to prove my identity when pushing or pulling code. I'm using one in this project because GitHub no longer allows password-based authentication for Git over HTTPS, and tokens provide a safer and more reliable way to connect my local repo with my GitHub repository.

I set up a GitHub token by going into my GitHub account settings, selecting Developer settings > Personal access tokens >Tokens (classic), and then generating a new token with the necessary scopes (e.g. repo for full repository access). After generating it, I copied the token immediately (since it can't be viewed again later) and used it in place of my password when Git prompted me during push or pull commands.

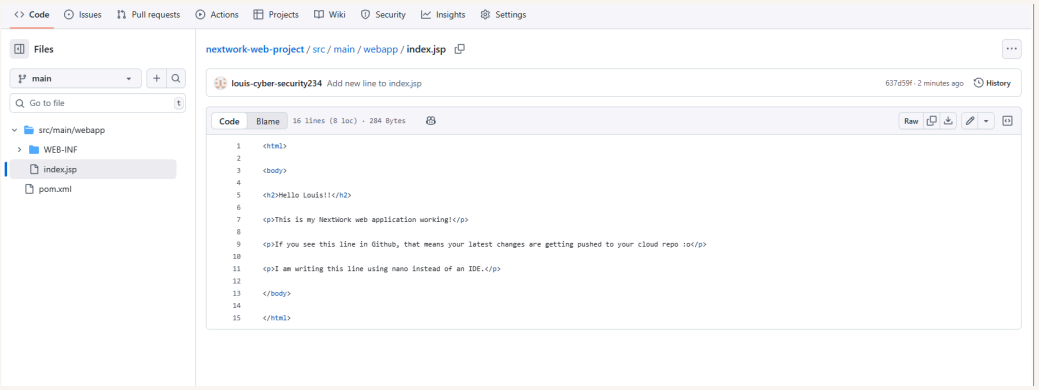

New personal access token (classic)
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.
Note
Generated for EC2 Instance Access. This is a part of NextWork's 7 D
What's this token for?
Expiration
30 days (Sep 26, 2025)
The token will expire on the selected date
Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo
Full control of private repositories
repo:status
Access commit status
repo_deployment
Access deployment status
public_repo
Access public repositories
repo:invite
Access repository invitations
security_events
Read and write security events
workflow
Update GitHub Action workflows
write:packages
Upload packages to GitHub Package Registry
read:packages
Download packages from GitHub Package Registry
delete:packages
Delete packages from GitHub Package Registry
admin:org
Full control of orgs and teams, read and write org projects

Louis Moyo
NextWork Student

Making changes again

I wanted to see Git working in action, so I updated the index.jsp file inside src/main/webapp. I couldn't see the changes in my GitHub repo initially because I hadn't staged and committed the file correctly, and at one point the folder was being treated like a submodule instead of a normal tracked file.

I finally saw the changes in my GitHub repo after staging the file with git add, committing it with git commit -m "message", and then pushing it to the remote repository using git push.



Setting up a README file

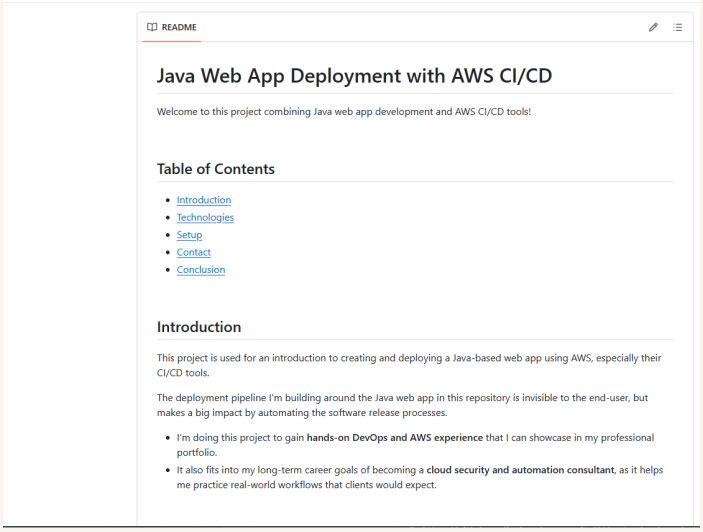
As a finishing touch to my GitHub repository, I added a README file, which is a simple markdown file (README.md) that introduces and documents the project. A README usually contains an overview of what the project does, instructions on how to set it up or run it, and any other important details like technologies used or future improvements. I added a README file by creating a new file called README.md in my project's root folder, writing my documentation in markdown format, and then committing and pushing it to GitHub. This way, when someone visits my repository, the README file is displayed automatically on the repo's homepage, making the project clearer and more professional.

My README is written in Markdown because it's a simple lightweight language that lets me create clean documentation without needing complex formatting tools. Markdown makes it easy to structure content in a way that looks professional on GitHub. Special characters can help format text in Markdown, such as: # for headings * or _ for italic and bold text - or * for bullet points ` for inline code triple backticks (```) for code blocks []() for hyperlinks This allows me to present my project clearly, with sections like Introduction, Technologies, Setup, and Contact, making it easier for others (and myself later) to understand the project.

placeholder

My README file has 6 sections that outline the key parts of my project. It starts with an Introduction explaining the purpose of the project and why I'm doing it. The Technologies section lists the tools I'm using (like GitHub, AWS, and VS Code) along with some challenges I faced and how I solved them. The Setup section provides step-by-step instructions and troubleshooting tips for anyone who wants to run the project themselves. In Contact, I've added my professional details, LinkedIn profile, and a placeholder for a profile photo so others can connect with me.

The Conclusion section summarises what I've learned and how this project helps my career goals. Finally, I included a Table of Contents at the top to make the README easy to navigate.



README

Java Web App Deployment with AWS CI/CD

Welcome to this project combining Java web app development and AWS CI/CD tools!

Table of Contents

Introduction

This project is used for an introduction to creating and deploying a Java-based web app using AWS, especially their CI/CD tools.

The deployment pipeline I'm building around the Java web app in this repository is invisible to the end-user, but makes a big impact by automating the software release processes.

- I'm doing this project to gain **hands-on DevOps and AWS experience** that I can showcase in my professional portfolio.
- It also fits into my long-term career goals of becoming a **cloud security and automation consultant**, as it helps me practice real-world workflows that clients would expect.

The place to learn & showcase your skills

Check out nextwork.org for more projects