



NextWork.org

Connect a GitHub Repo with AWS

G Gloria

```
Installing : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64          4/8
Installing : perl-File-Find-1.37-477.amzn2023.0.6.noarch           5/8
Installing : perl-Error-1:0.17029-5.amzn2023.0.2.noarch           6/8
Installing : perl-Git-2.40.1-1.amzn2023.0.3.noarch             7/8
Installing : git-2.40.1-1.amzn2023.0.3.x86_64                  8/8
Running scriptlet: git-2.40.1-1.amzn2023.0.3.x86_64           8/8
Verifying   : git-2.40.1-1.amzn2023.0.3.x86_64                  1/8
Verifying   : git-core-2.40.1-1.amzn2023.0.3.x86_64            2/8
Verifying   : git-core-doc-2.40.1-1.amzn2023.0.3.noarch         3/8
Verifying   : perl-Error-1:0.17029-5.amzn2023.0.2.noarch         4/8
Verifying   : perl-File-Find-1.37-477.amzn2023.0.6.noarch         5/8
Verifying   : perl-Git-2.40.1-1.amzn2023.0.3.noarch           6/8
Verifying   : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64          7/8
Verifying   : perl-lib-0.65-477.amzn2023.0.6.x86_64           8/8

Installed:
git-2.40.1-1.amzn2023.0.3.x86_64      git-core-2.40.1-1.amzn2023.0.3.x86_64      git-core-doc-2.40.1-1.amzn2023.0.3.noarch
perl-Error-1:0.17029-5.amzn2023.0.2.noarch perl-File-Find-1.37-477.amzn2023.0.6.noarch perl-Git-2.40.1-1.amzn2023.0.3.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
[ec2-user@ip-172-31-18-36 ~]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-18-36 ~]$ 
```



Introducing Today's Project!

What is GitHub?

GitHub is an online platform for storing and managing code using Git. I used GitHub to create a repository, connect it to my web app, and push changes. This helped me track updates, save my code, and synchronize it securely in the cloud.

One thing I didn't expect...

One thing I didn't expect in this project was the need to use a GitHub token for authentication instead of a password. This step highlighted GitHub's focus on security and gave me a better understanding of modern authentication practices.

This project took me...

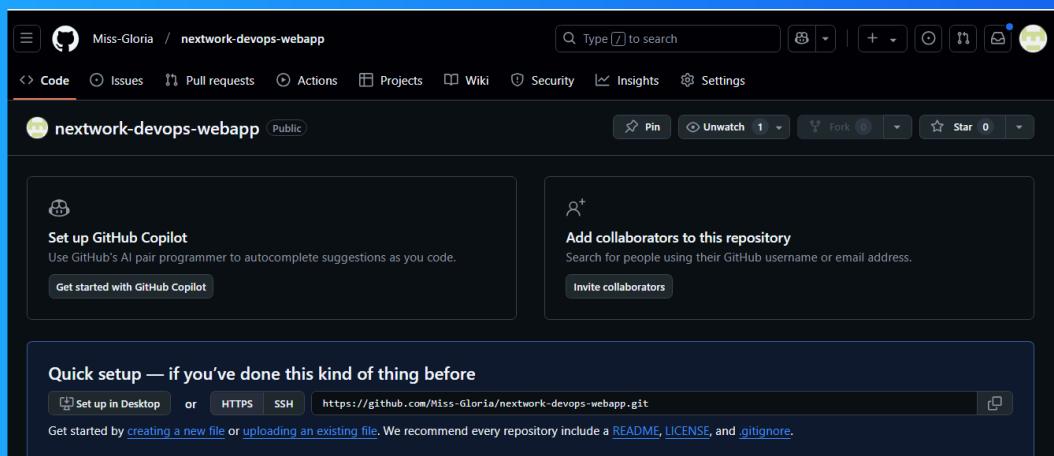
This project took me about 20 minutes to complete as it was a continuation of the previous project. Building on the earlier setup made the process quicker and more seamless, allowing me to focus on learning Git and GitHub integration.



Git and GitHub

Git is a version control tool that tracks code changes and helps manage collaboration. I installed Git using the commands `sudo dnf update -y` to update the system and `sudo dnf install git -y` to download and install Git on my EC2 instance.

GitHub is an online platform for storing and managing code repositories using Git. I'm using GitHub in this project to securely store my web app's source code, track changes, and enable collaboration, making it easier to manage and share the project.



My local repository

A Git repository is a storage space where your project's code and its change history are saved. It allows you to track, manage, and revert changes, making collaboration and version control easier. It can be on your computer or hosted on Github.

git init is a command that initializes a new Git repository in a folder, enabling version control for the project. I ran git init in my web app folder on the EC2 instance to start tracking changes and prepare it for connection to my GitHub repository.

After running git init, the terminal response showed that an empty Git repository was launched, with master as the default branch. A branch in Git is a model of the project where changes can be made separate, ensuring the main code remain unaffected.

```
● [ec2-user@ip-172-31-18-36 nextwork-web-project]$ 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>
Initialized empty Git repository in /home/ec2-user/nextwork-web-project/.git/
○ [ec2-user@ip-172-31-18-36 nextwork-web-project]$ []
```



To push local changes to GitHub, I ran three commands

git add

The first command I ran was `git add .`, which stages all modified files in my project. A staging area is where Git collects changes for review before committing them. This step ensures I can check for mistakes or unwanted edits before saving them.

git commit

The second command I ran was `git commit -m Updated index.jsp` with new content. Using `-m` means I added a message describing the changes. This helps track and understand updates in the project's version history, making it easier to review changes later.

git push

The third command I ran was `git push -u origin master`. Using `-u` means I set an upstream branch, so Git remembers to push changes to the specified GitHub repository (`origin`) and branch (`master`). This simplifies future pushes, making it simple and fast.



Authentication

When I commit changes to GitHub, Git asks for my credentials because it needs to verify that I'm authorized to make changes to the repository. This ensures security by confirming that only authorized users can push updates to the GitHub repository.

Local Git identity

Git needs my name and email because they are used to identify the author of each commit. This information is added to the project's history, helping collaborators track who made changes and when, ensuring clear and organized version control.

Running `git log` showed me the commit history, including a unique commit ID, the author's name and email, the date of the commit, and the commit message. This information helps track changes and provides a clear history of updates in the project.

```
● [ec2-user@ip-172-31-18-36 nextwork-web-project]$ git log
commit 75229fccd0686c87f967b8c0026aa6fb0f651421 (HEAD -> master, origin/master)
Author: EC2 Default User <ec2-user@ip-172-31-18-36.ec2.internal>
Date:   Mon Dec 23 21:55:25 2024 +0000

    Updated index.jsp with new content
○ [ec2-user@ip-172-31-18-36 nextwork-web-project]$ █
```



GitHub tokens

GitHub authentication failed when I entered my password because GitHub no longer supports password authentication for Git operations. Instead, it requires the use of a personal access token for secure authentication when pushing or pulling changes.

A GitHub token is a secure, personal access key used to authenticate with GitHub instead of a password. I'm using one in this project because GitHub no longer supports passwords for operation, and it ensures secure and seamless access to my repo.

I could set up a GitHub token by going to my GitHub account settings, selecting Developer settings, then Personal access tokens, and clicking Generate new token. I chose the necessary permissions, generated the token, and copied it for Git operations.

The screenshot shows the GitHub 'New personal access token (classic)' creation interface. The page has a dark background with blue highlights. At the top, it says 'New personal access token (classic)'. Below that, a note states: '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](#)'. A 'Note' section contains a note about generating tokens for EC2 Instance Access for the NextWork DevOps project series. It asks 'What's this token for?' and shows an 'Expiration *' dropdown set to '30 days' with a note that the token will expire on Wed, Jan 22 2025. A 'Select scopes' section is present with a note explaining that scopes define access for personal tokens and a link to 'Read more about OAuth scopes'.



Making changes again

I wanted to see Git working, so I updated the index.jsp file in the nextwork-web-project. I couldn't see the changes in my GitHub repo at first because the updates had to be committed locally and then pushed to the GitHub repository to appear there.

I finally saw the changes in my GitHub repo after committing the updates to the index.jsp file locally using git commit and then pushing them to the remote repository with git push. This ensured the changes were saved and reflected in the GitHub repo

The screenshot shows a GitHub code editor interface for the file `index.jsp`. The file contains the following JavaServer Pages (JSP) code:

```
<html>
<body>
<h2>Hello Gloria</h2>
<p>This is my NextWork web application working!</p>
<p>If you see this line in Github, that means your latest changes are getting pushed to your cloud repo :o</p>
</body>
</html>
```

The code editor includes standard GitHub features like a blame history, raw view, and download options.



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

