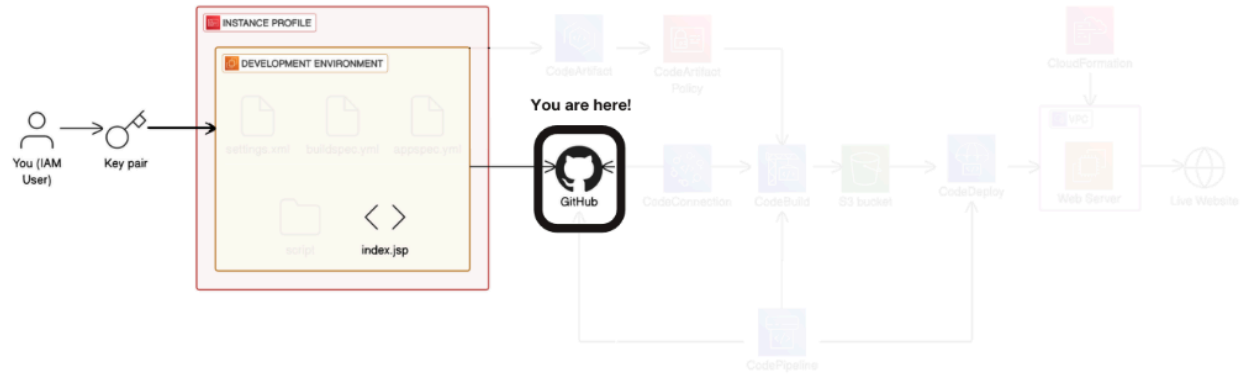
Project 2: Connect a Github Repo with AWS

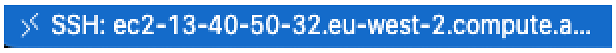


Project objectives:

- Set up Git and GitHub.
- Connect your web app project to a GitHub repo.
- Make changes to your web app files and watch your GitHub repo update too.

Step 1: Refer to Project 1: Set up a web app in AWS

Launch your EC2 instance in VS Code (screenshot below shows Remote SSH connection to our EC2 instance via VS Code)

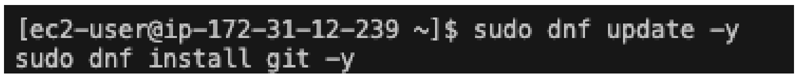


```
SSH: ec2-13-40-50-32.eu-west-2.compute.a...
```

Step 2: Install Git

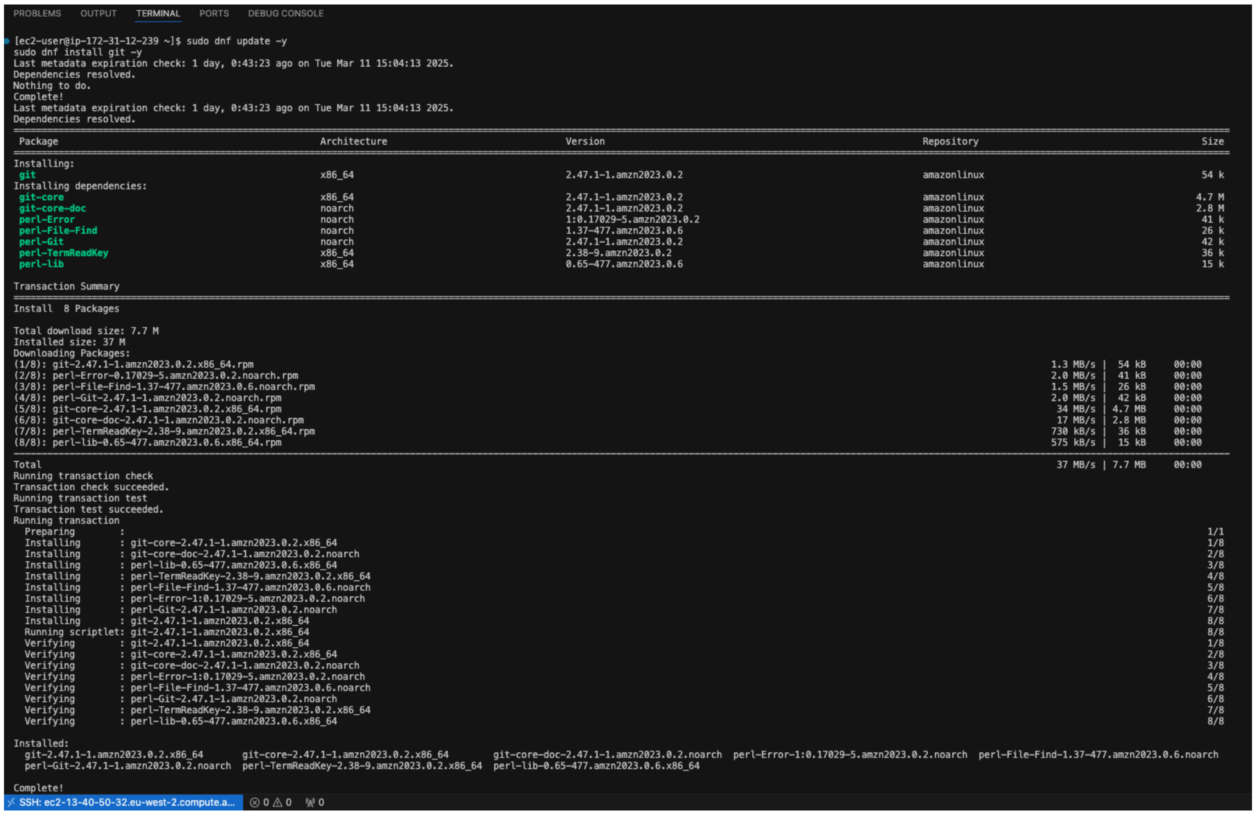
In this step we are installing Git in our EC2 instance terminal. Git is a version control system that tracks changes made by who and when.

Use the below Bash code to install Git in your EC2 instance. Remember to use the EC2 instance terminal.



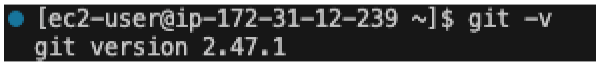
```
[ec2-user@ip-172-31-12-239 ~]$ sudo dnf update -y
sudo dnf install git -y
```

Below is the result of a complete installation:



```
[ec2-user@ip-172-31-12-239 ~]$ sudo dnf update -y
sudo dnf install git -y
Last metadata expiration check: 1 day, 0:43:23 ago on Tue Mar 11 15:04:13 2025.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 1 day, 0:43:23 ago on Tue Mar 11 15:04:13 2025.
Dependencies resolved.
==========================================================================================================================
 Package                     Architecture         Version                          Repository                         Size
==========================================================================================================================
Installing:
 git                         x86_64               2.47.1-1.amzn2023.0.2            amazonlinux                        54 k
Installing dependencies:
 git-core                    x86_64               2.47.1-1.amzn2023.0.2            amazonlinux                       4.7 M
 git-core-doc                noarch               2.47.1-1.amzn2023.0.2            amazonlinux                       2.8 M
 perl-Error                  noarch               1:0.17029-5.amzn2023.0.2         amazonlinux                        41 k
 perl-File-Find              noarch               1.37-477.amzn2023.0.6            amazonlinux                        26 k
 perl-Git                    noarch               2.47.1-1.amzn2023.0.2            amazonlinux                        42 k
 perl-TermReadKey            x86_64               2.38-9.amzn2023.0.2              amazonlinux                        36 k
 perl-lib                    x86_64               0.65-477.amzn2023.0.6            amazonlinux                        15 k

Transaction Summary
==========================================================================================================================
Install  8 Packages

Total download size: 7.7 M
Installed size: 37 M
Downloading Packages:
(1/8): git-2.47.1-1.amzn2023.0.2.x86_64.rpm                                         1.3 MB/s |  54 kB     00:00
(2/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm                                 2.0 MB/s |  41 kB     00:00
(3/8): perl-File-Find-1.37-477.amzn2023.0.6.noarch.rpm                              1.5 MB/s |  26 kB     00:00
(4/8): perl-Git-2.47.1-1.amzn2023.0.2.noarch.rpm                                    2.0 MB/s |  42 kB     00:00
(5/8): git-core-2.47.1-1.amzn2023.0.2.x86_64.rpm                                     34 MB/s | 4.7 MB     00:00
(6/8): git-core-doc-2.47.1-1.amzn2023.0.2.noarch.rpm                                 17 MB/s | 2.8 MB     00:00
(7/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm                              730 kB/s |  36 kB     00:00
(8/8): perl-lib-0.65-477.amzn2023.0.6.x86_64.rpm                                    575 kB/s |  15 kB     00:00
--------------------------------------------------------------------------------------------------------------------------
Total                                                                                37 MB/s | 7.7 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                   1/1
  Installing       : git-core-2.47.1-1.amzn2023.0.2.x86_64                                                             1/8
  Installing       : git-core-doc-2.47.1-1.amzn2023.0.2.noarch                                                         2/8
  Installing       : perl-lib-0.65-477.amzn2023.0.6.x86_64                                                             3/8
  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-2.47.1-1.amzn2023.0.2.noarch                                                             7/8
  Installing       : git-2.47.1-1.amzn2023.0.2.x86_64                                                                  8/8
  Running scriptlet: git-2.47.1-1.amzn2023.0.2.x86_64                                                                  8/8
  Verifying        : git-2.47.1-1.amzn2023.0.2.x86_64                                                                  1/8
  Verifying        : git-core-2.47.1-1.amzn2023.0.2.x86_64                                                             2/8
  Verifying        : git-core-doc-2.47.1-1.amzn2023.0.2.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-2.47.1-1.amzn2023.0.2.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-2.47.1-1.amzn2023.0.2.x86_64       git-core-2.47.1-1.amzn2023.0.2.x86_64       git-core-doc-2.47.1-1.amzn2023.0.2.noarch  perl-Error-1:0.17029-5.amzn2023.0.2.noarch  perl-File-Find-1.37-477.amzn2023.0.6.noarch
  perl-Git-2.47.1-1.amzn2023.0.2.noarch  perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64  perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
```

Verify the installation by checking the version of Git that has been installed:



```
[ec2-user@ip-172-31-12-239 ~]$ git -v
git version 2.47.1
```

Step 3: Set up GitHub

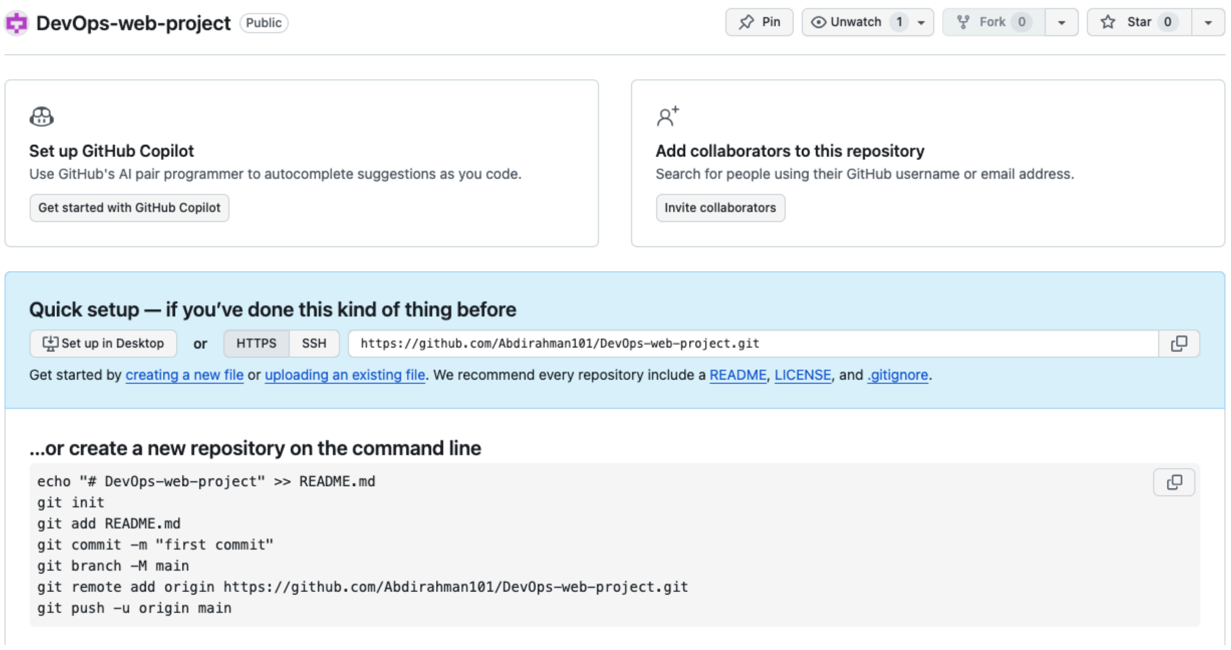
In this step, we are going to:

- Create a Github repository

Since I have already got a GitHub account, we will only need to create a GitHub repository. GitHub is a storage space that allows for different versions of your project that Git tracks. Git is the tool that actually tracks the changes.

<u>Create the GitHub repo:</u>

A repository (repo) is essentially a folder that can store all project files and their history. Since GitHub is hosted on the cloud, we can allow collaboration with other developers and engineers.


DevOps-web-project Public

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/Abdirahman101/DevOps-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 "# DevOps-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/Abdirahman101/DevOps-web-project.git
git push -u origin main
```

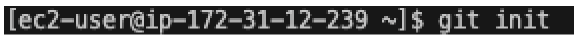
Step 4: Commit and Push your changes to GitHub

In this step, we are going to:

- Set up a local git repo in your web app folder.
- Connect your local repo with your GitHub repo.

We now have our GitHub repo created and Git installed on our EC2 instance. Next, we need to set up a local git repo in our web app folder.

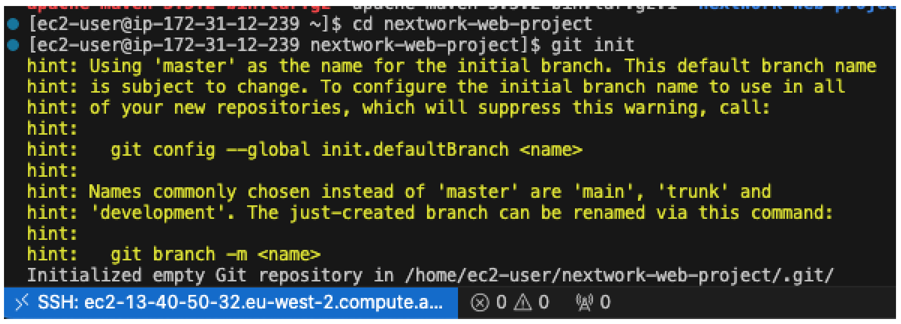
Open the EC2 instance terminal and put the following command in:



```
[ec2-user@ip-172-31-12-239 ~]$ git init
```

'git init' is used to initialise/set up a local repository within our EC2 instance. We ran git init in our web app folder, this tells our terminal that we want to track changes locally.

The result of running the above command is shown below:

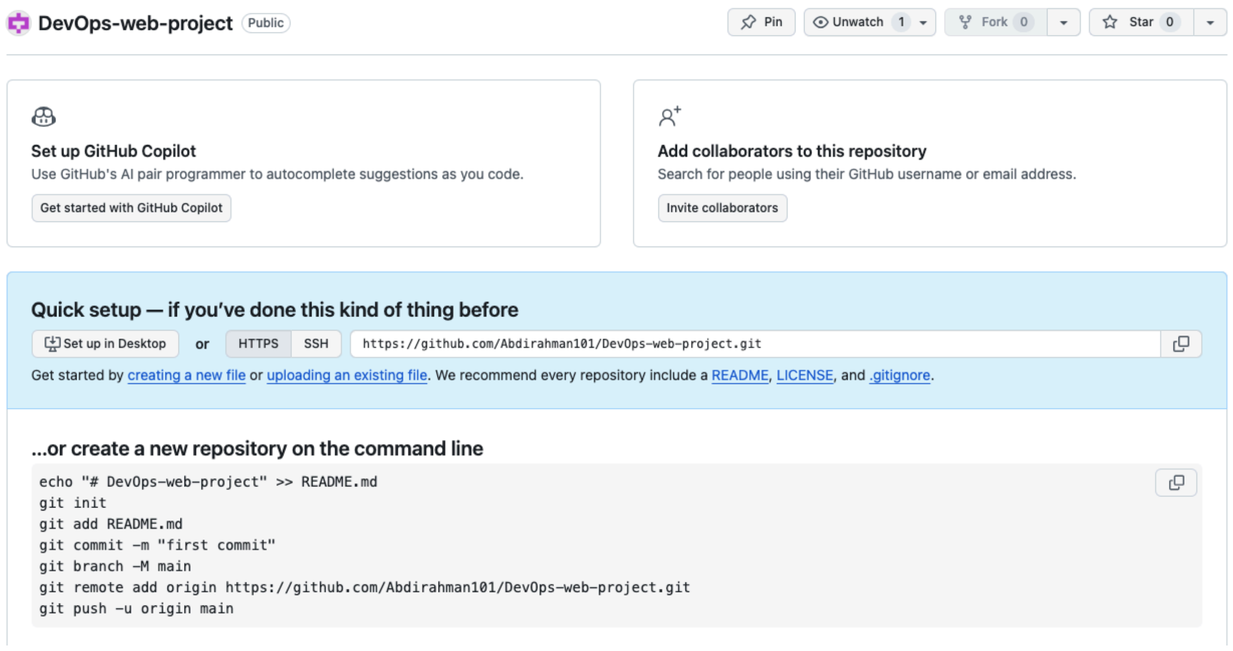


```
[ec2-user@ip-172-31-12-239 ~]$ cd nextwork-web-project
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, 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/
```

The yellow text lets me know that I have created a local repository within our EC2 instance and the initial branch will be named ‘master’. You can create new branches in which you can experiment (trial and error) different methods without impacting the ‘master’ branch. It is best practice to make changes in a separate branch and then merge the changes into the master once they're ready. Today, we will just use the ‘master’ branch and commit all changes to that as I don't have many changes to commit.

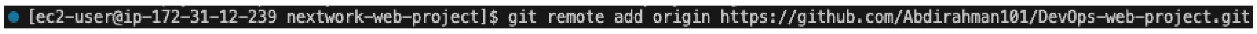
Connect local repo with GitHub:

Return to the GitHub repo and copy the HTML URL generated (when the repo was created earlier):


DevOps-web-project
Public
Pin
Unwatch 1
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/Abdirahman101/DevOps-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 "# DevOps-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/Abdirahman101/DevOps-web-project.git
git push -u origin main
```

Copy the HTML URL and use the below bash code in our EC2 instance terminal:



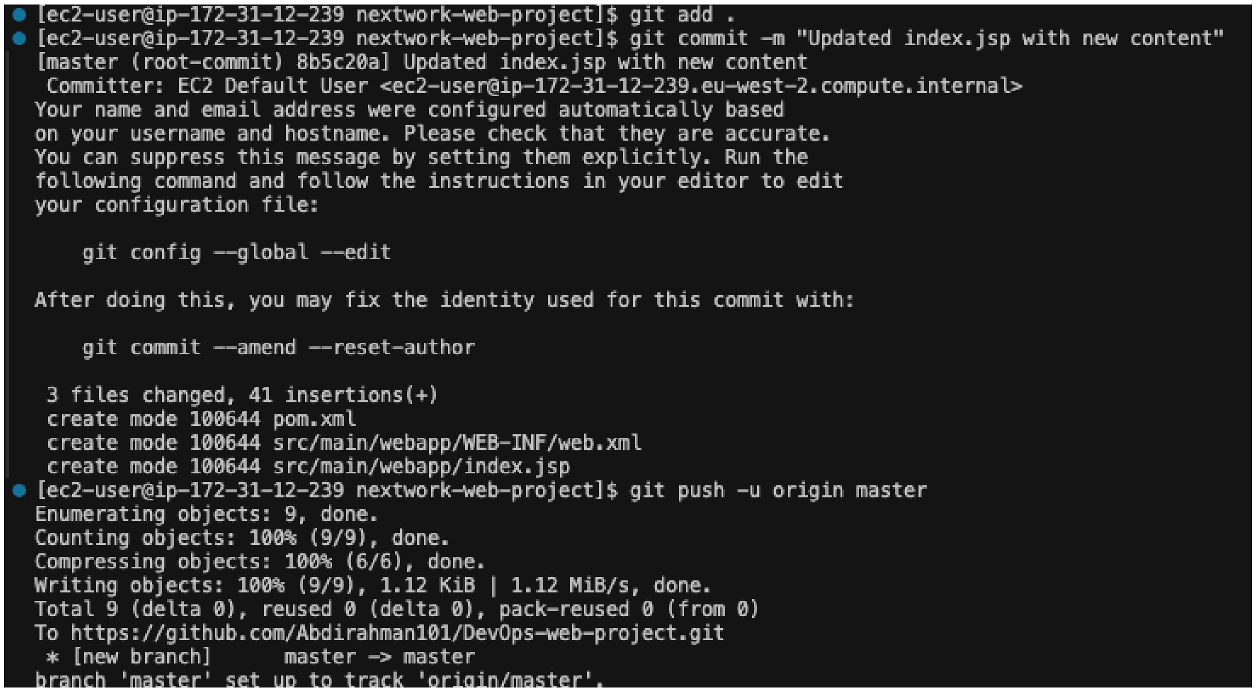
```
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git remote add origin https://github.com/Abdirahman101/DevOps-web-project.git
```

‘Remote add origin’ notifies Git where our GitHub repo is located.

Next run the command ‘Git add . ‘ to stage the changes which tells Git I am ready to put my modified files for review.

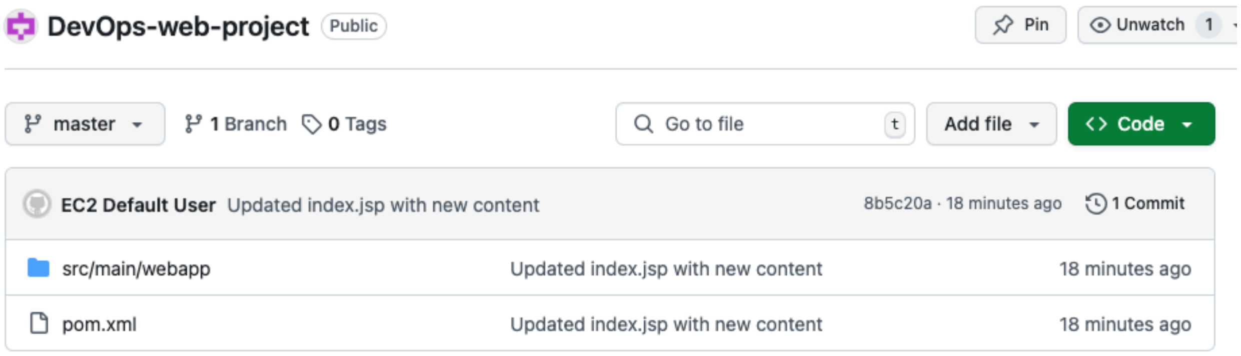
We can save the changes we have made by running the command ‘git commit -m “Updated index.jsp with new content”’. Using -m allows me to flag a message which I have added in the command.

Lastly we can now push our changes to GitHub repo. This is done using the command ‘git push -u origin master’. This tells git to push changes onto the origin (i.e. our GitHub repo) and to our master branch (i.e. changes should be reflected in our master branch). Using -u tells git to set up an upstream (i.e. reminder to git to remember to push to the master branch by default, so next time we can run ‘git push’ without needing to define our origin or master).



```
● [ec2-user@ip-172-31-12-239 nextwork-web-project]$ git add .
● [ec2-user@ip-172-31-12-239 nextwork-web-project]$ git commit -m "Updated index.jsp with new content"
  [master (root-commit) 8b5c20a] Updated index.jsp with new content
   Committer: EC2 Default User <ec2-user@ip-172-31-12-239.eu-west-2.compute.internal>
  Your name and email address were configured automatically based
  on your username and hostname. Please check that they are accurate.
  You can suppress this message by setting them explicitly. Run the
  following command and follow the instructions in your editor to edit
  your configuration file:

      git config --global --edit

  After doing this, you may fix the identity used for this commit with:

      git commit --amend --reset-author

   3 files changed, 41 insertions(+)
   create mode 100644 pom.xml
   create mode 100644 src/main/webapp/WEB-INF/web.xml
   create mode 100644 src/main/webapp/index.jsp
● [ec2-user@ip-172-31-12-239 nextwork-web-project]$ git push -u origin master
  Enumerating objects: 9, done.
  Counting objects: 100% (9/9), done.
  Compressing objects: 100% (6/6), done.
  Writing objects: 100% (9/9), 1.12 KiB | 1.12 MiB/s, done.
  Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
  To https://github.com/Abdirahman101/DevOps-web-project.git
   * [new branch]      master -> master
  branch 'master' set up to track 'origin/master'.
```

Now that we have successfully pushed changes to our repo, return to GitHub and you will now see your web app files along with the commit message that was written earlier.

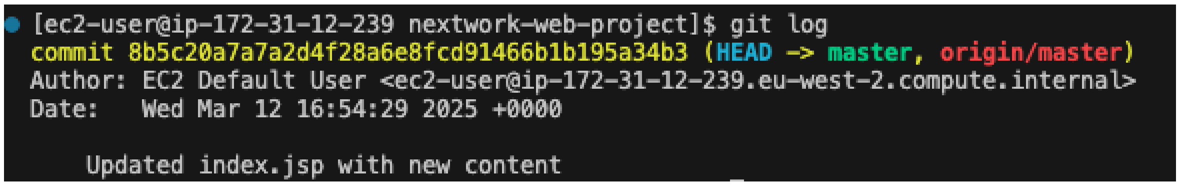


DevOps-web-project Public

EC2 Default User Updated index.jsp with new content	8b5c20a · 18 minutes ago	1 Commit
src/main/webapp	Updated index.jsp with new content	18 minutes ago
pom.xml	Updated index.jsp with new content	18 minutes ago

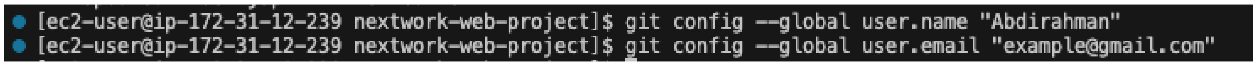
Git needs author information for commits to track who made what change. If you don't set it manually, Git uses the system's default username, which might not accurately represent your identity in your project's version history.

Run the command ‘git log’ in the EC2 terminal.



```
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git log
commit 8b5c20a7a7a2d4f28a6e8fcd91466b1b195a34b3 (HEAD -> master, origin/master)
Author: EC2 Default User <ec2-user@ip-172-31-12-239.eu-west-2.compute.internal>
Date:   Wed Mar 12 16:54:29 2025 +0000

    Updated index.jsp with new content
```

As you can see, the author is ‘EC2 Default User’ and the EC2 instance’s IPv4 DNS is not my email. To update this we run the commands shown below.



```
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git config --global user.name "Abdirahman"
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git config --global user.email "example@gmail.com"
```

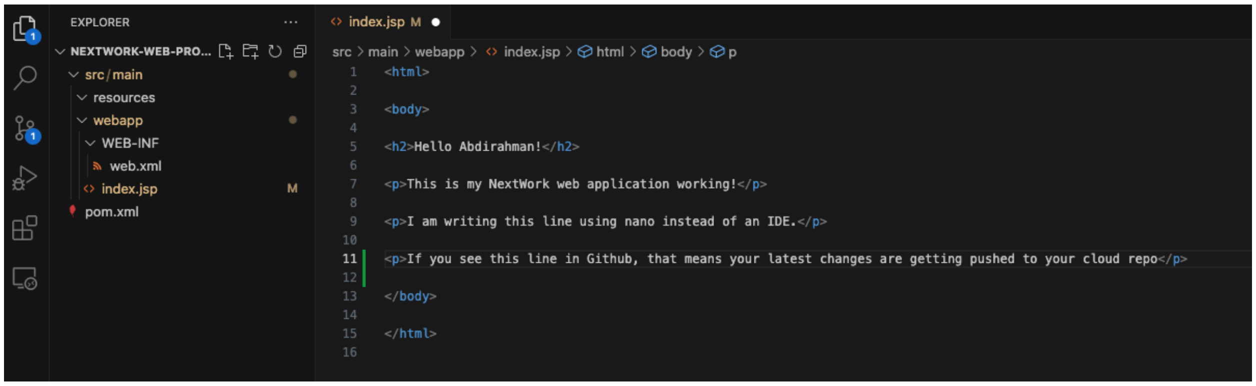
Step 5: Our Second Commit

In this step, we are going to:

- Make changes to our web app
- Commit and push those changes

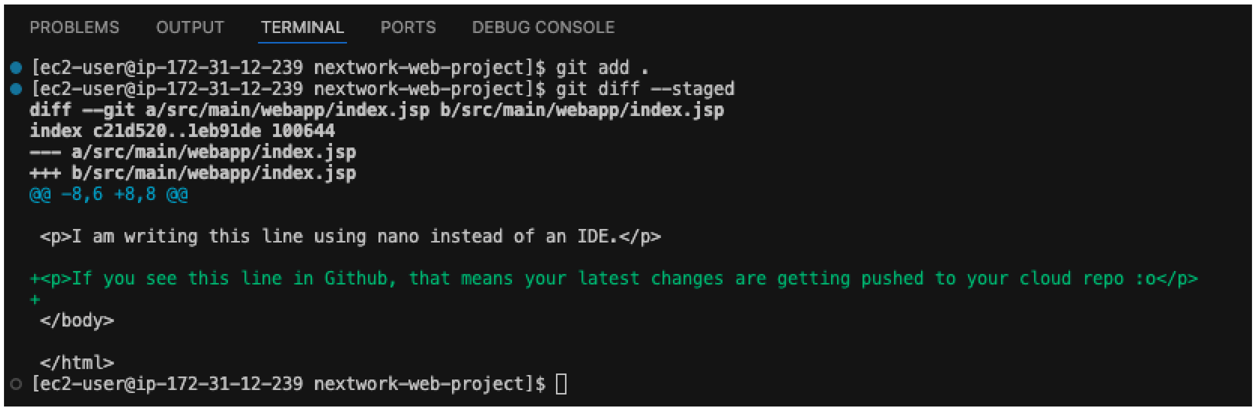
Make changes to our web app:

Open your EC2 instances VS Code window and open the web app folder. Add a new paragraph to the index.jsp file stating: “If you see this line in Github, that means your latest changes are getting pushed to your cloud repo”. Ensure to save changes by pressing Cmd + S.



```
<html>

<body>

<h2>Hello Abdirahman!</h2>

<p>This is my NextWork web application working!</p>

<p>I am writing this line using nano instead of an IDE.</p>

<p>If you see this line in Github, that means your latest changes are getting pushed to your cloud repo</p>

</body>

</html>
```

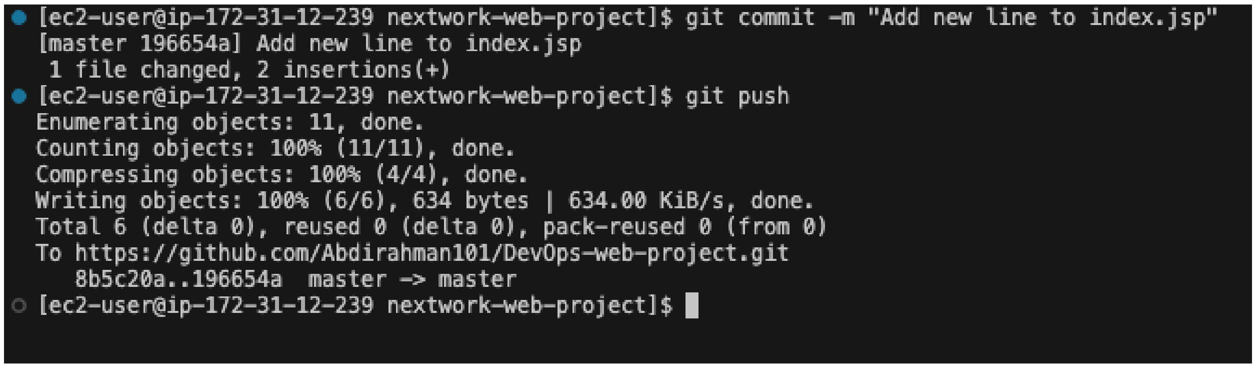
Head back to the EC2 instance terminal and run the command ‘git add .’ again to stage a change. Then run then command ‘git diff --staged’ to show us the exact changes have been staged compared to the last commit.



```
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git add .
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git diff --staged
diff --git a/src/main/webapp/index.jsp b/src/main/webapp/index.jsp
index c21d520..1eb91de 100644
--- a/src/main/webapp/index.jsp
+++ b/src/main/webapp/index.jsp
@@ -8,6 +8,8 @@

 <p>I am writing this line using nano instead of an IDE.</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>
[ec2-user@ip-172-31-12-239 nextwork-web-project]$
```

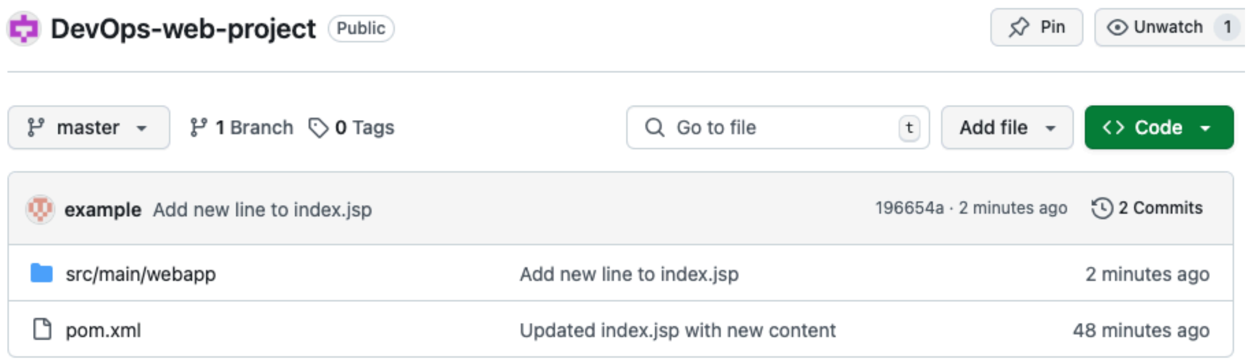
Commit and push changes:

To commit our changes, run the command ‘git commit -m “Add new line to index.jsp”. To push the changes run the command ‘git push’. Notice how we don't need to include -u origin master anymore.

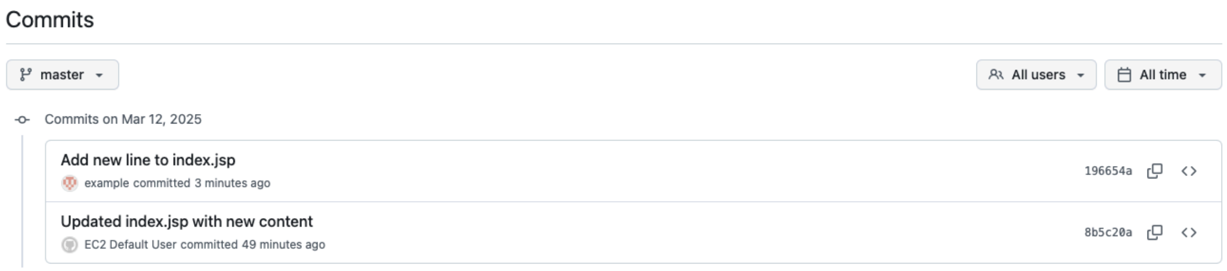


```
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git commit -m "Add new line to index.jsp"
[master 196654a] Add new line to index.jsp
 1 file changed, 2 insertions(+)
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 634 bytes | 634.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Abdirahman101/DevOps-web-project.git
   8b5c20a..196654a  master -> master
[ec2-user@ip-172-31-12-239 nextwork-web-project]$
```

Head back to the GitHub repo and now we can see that a change was made 2 minutes ago and a new commit was added.



example Add new line to index.jsp		196654a · 2 minutes ago
src/main/webapp	Add new line to index.jsp	2 minutes ago
pom.xml	Updated index.jsp with new content	48 minutes ago

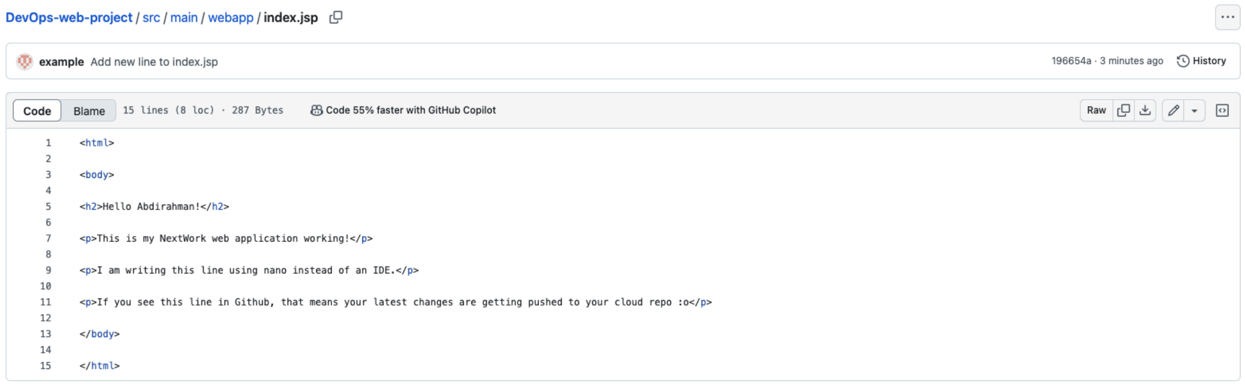

Commits

master · All users · All time

Commits on Mar 12, 2025

Add new line to index.jsp
example committed 3 minutes ago — 196654a

Updated index.jsp with new content
EC2 Default User committed 49 minutes ago — 8b5c20a


DevOps-web-project / src / main / webapp / index.jsp

example Add new line to index.jsp — 196654a · 3 minutes ago — History

Code · Blame · 15 lines (8 loc) · 287 Bytes · Code 55% faster with GitHub Copilot · Raw

```
<html>

<body>

<h2>Hello Abdirahman!</h2>

<p>This is my NextWork web application working!</p>

<p>I am writing this line using nano instead of an IDE.</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>
```

Summary:

1. **Set up a GitHub repository:** You created a new repository in AWS GitHub to securely store and manage the source code for your Java web app.
2. **Configure Git and a local repository:** You established your Git identity with your username and email. You also initialized a local repo with your GitHub repo as the remote origin.
3. **Make Your first commit and push:** You added all your files to the staging area, committed them, and pushed these changes to the master branch of your GitHub repository, making your code available in the cloud.

Next up, we need to find a way to store our web app's **packages** and **dependencies**, which are pieces of code your web app relies on in order to work. This is where **AWS CodeArtifact** comes into play.