😊 Connecting GitHub account to AWS Developer Tools

GitHub is a widely used web-based platform that facilitates software development and version control using Git. It provides a collaborative environment where developers can work together on projects, manage code, and track changes. Here's a comprehensive overview:

Key Components and Features

1. **Version Control with Git:**
    - GitHub is built on Git, a distributed version control system that tracks changes in source code. This allows multiple developers to work on a project simultaneously without conflicts.
    - Git keeps a complete history of changes, enabling developers to revert to previous versions if necessary.
2. **Repositories (Repos):**
    - A repository is a centralized location where all the files related to a project are stored. Each repository contains the project's source code, documentation, and other resources.
    - Repositories can be public (accessible to everyone) or private (restricted access).
3. **Branches:**
    - Branches are parallel versions of a repository. They allow developers to work on new features, bug fixes, or experiments without affecting the main codebase (often called the "main" or "master" branch).
    - Once the work on a branch is complete, it can be merged back into the main branch.
4. **Pull Requests (PRs):**
    - Pull requests are a key feature for collaboration. They are used to propose changes from one branch to another. The proposed changes can be reviewed, discussed, and approved before being merged into the main branch.
    - Pull requests facilitate code review and help maintain code quality.
5. **Issues and Bug Tracking:**
    - GitHub provides an integrated issue tracker to manage bugs, tasks, and feature requests. Users can create, comment on, and track the status of issues.
    - This feature helps teams organize and prioritize work.
6. **Collaborative Features:**

    - GitHub supports collaboration through features like forking (creating a personal copy of someone else's repository), code review, and discussions.
    - Team members can comment on code, discuss changes, and provide feedback directly within the platform.
7. **GitHub Pages:**
    - GitHub Pages is a service that allows users to host static websites directly from a GitHub repository. It's commonly used for project documentation, personal portfolios, and more.
8. **Continuous Integration/Continuous Deployment (CI/CD):**
    - GitHub integrates with various CI/CD tools, enabling automated testing, building, and deployment of projects. This helps ensure code quality and accelerates the development process.
9. **Security and Insights:**
    - GitHub offers security features like vulnerability alerts, secret scanning, and dependency management.
    - Insights and analytics tools provide metrics on project activity, code changes, and collaboration.
10. **Community and Open Source:**
    - GitHub is a central hub for open-source projects. Developers can contribute to projects, collaborate with others, and discover new projects.
    - The platform hosts millions of repositories, making it a valuable resource for learning and sharing code.

Use Cases

- **Open-Source Development:** GitHub is a go-to platform for open-source projects, where contributors from around the world can collaborate.
- **Team Collaboration:** Organizations and teams use GitHub to manage internal projects, track issues, and maintain a streamlined workflow.
- **Documentation and Knowledge Sharing:** GitHub Pages and wikis are used for project documentation and knowledge sharing.

Access

GitHub is accessible through its website (github.com) and offers both free and paid plans. The free tier provides ample functionality for most users, while paid plans offer additional features like private repositories and advanced security options.

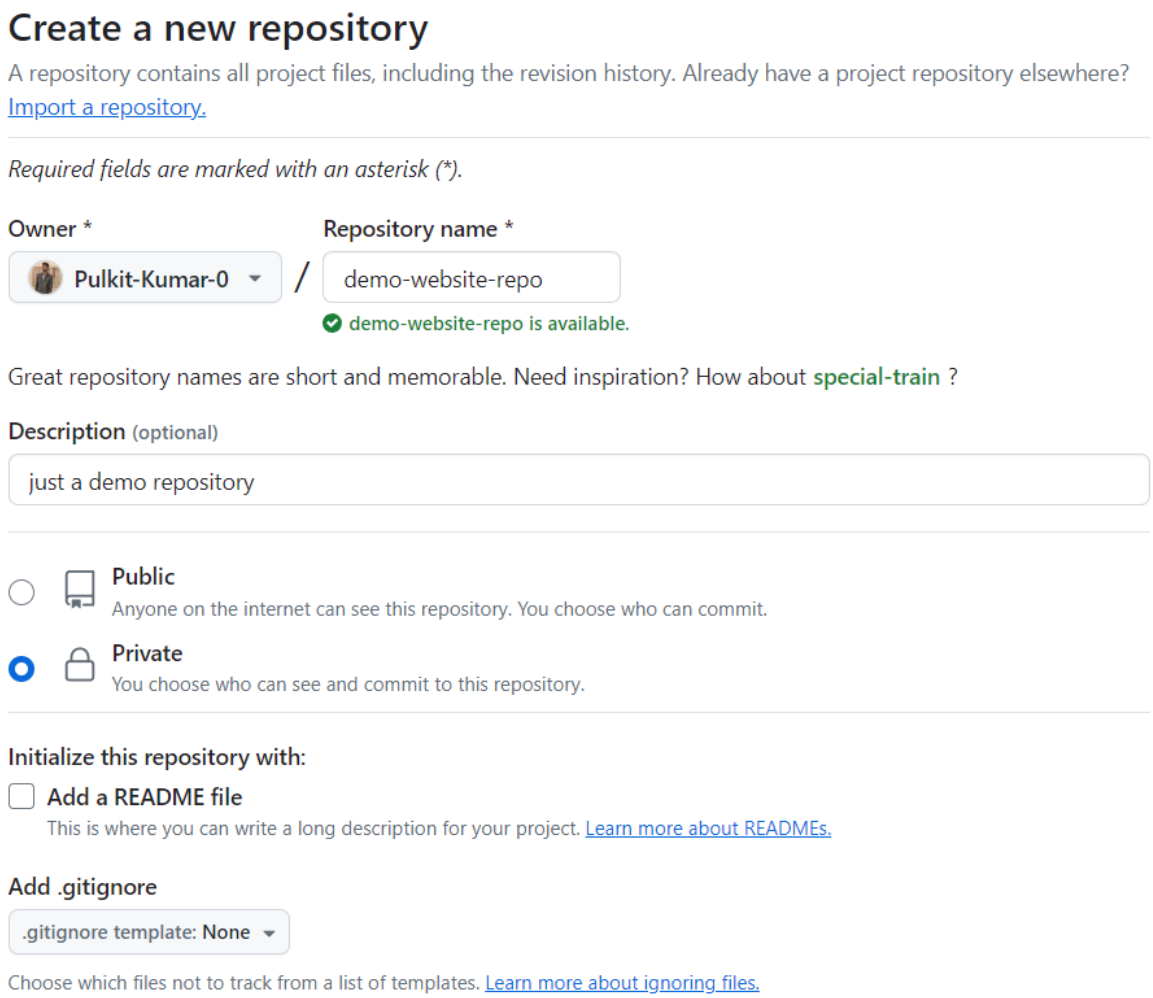
In summary, GitHub is a powerful platform for version control, collaboration, and project management, making it a cornerstone of modern software development practices.

End Goal: In this lab, you will learn how to connect a GitHub repository with AWS to automate the deployment of a website. The end goal is to establish a connection between GitHub and the AWS Developer tool and create a code pipeline so that whenever you update your website's code on GitHub, the changes are automatically applied to your live website without you having to do anything manually. This setup saves time and ensures your site always has the latest updates.

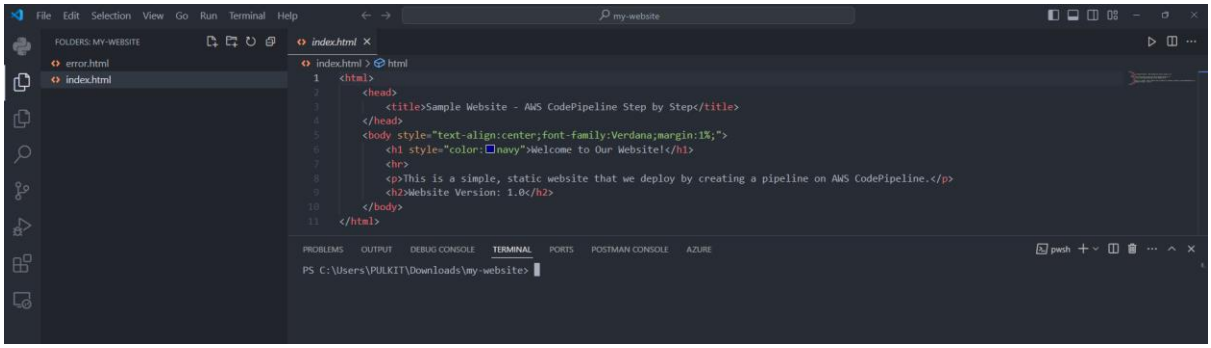
😄 To begin with the Lab:

😄 Step 1: Creating GitHub Repo

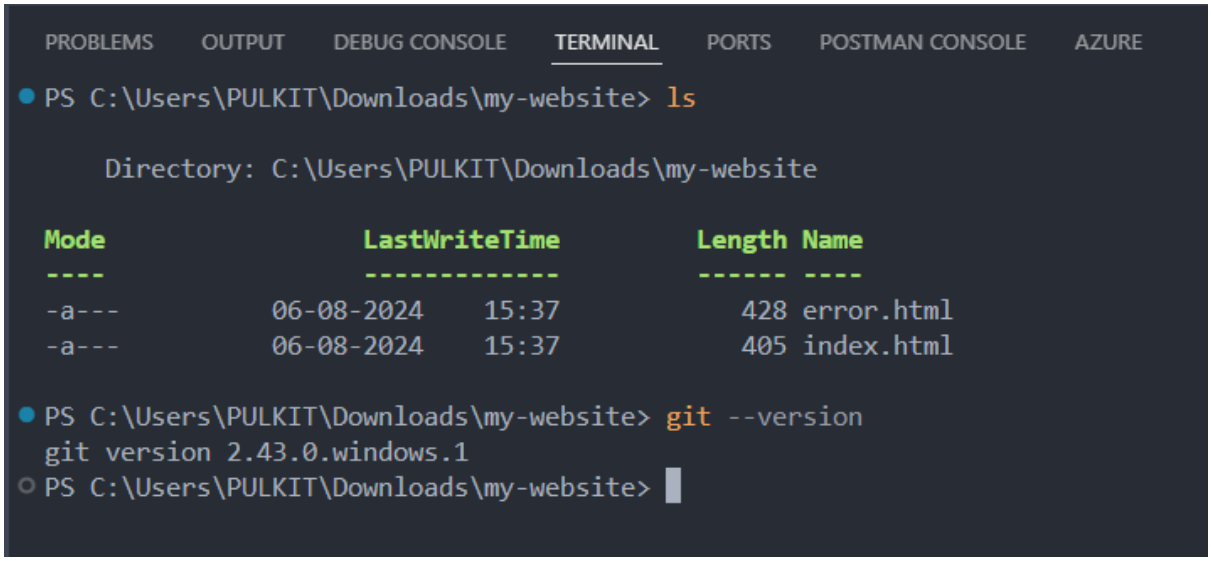
1. In this lab we will learn how to connect our GitHub account with AWS Developer Tools. So, for that first, we need to create a private repository in our GitHub account.



2. Now you will find a zipped folder on our GitHub account with the name my-website. You need to download it and then unzip the content then open the folder in VS Code. Also, open the terminal, as you can see below in the snapshot.



3. You need to check only these two files should be in your my-website folder and git should be installed on your local machine.



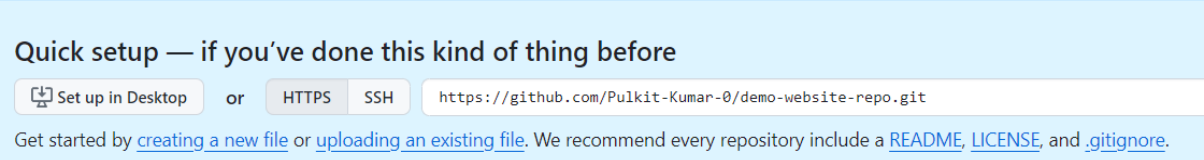
4. To begin with, you must initialize a local Git repository under this folder. It's straightforward. You can do this using the 'git init' command. After that, you can check the status of your local Git repository using the 'git status' command.
5. As you can see, none of our files have been tracked by Git. It means they won't be included in our commits. So next, we will stage our files using the 'git add' command, and we will provide '.' to select all files under this folder.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  AZURE

PS C:\Users\PULKIT\Downloads\my-website> git init
Initialized empty Git repository in C:/Users/PULKIT/Downloads/my-website/.git/
PS C:\Users\PULKIT\Downloads\my-website> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        error.html
        index.html

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\PULKIT\Downloads\my-website> git add .
warning: in the working copy of 'error.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\PULKIT\Downloads\my-website> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   error.html
        new file:   index.html

PS C:\Users\PULKIT\Downloads\my-website>
```

6. After providing the commit command you can see that our first git commit has been created.

```
PS C:\Users\PULKIT\Downloads\my-website> git commit -m "Version 1.0"
[master (root-commit) ee079cc] Version 1.0
 2 files changed, 22 insertions(+)
 create mode 100644 error.html
 create mode 100644 index.html
PS C:\Users\PULKIT\Downloads\my-website>
```

7. However, it currently only exists in our local Git repository. We must push it to our GitHub repository to be able to create a pipeline from it later. First, let's add our GitHub repository as a remote repository using the 'git remote add' command.
8. For that, first we need to copy the HTTPS URL from GitHub. You can get it from the quick setup as shown below.



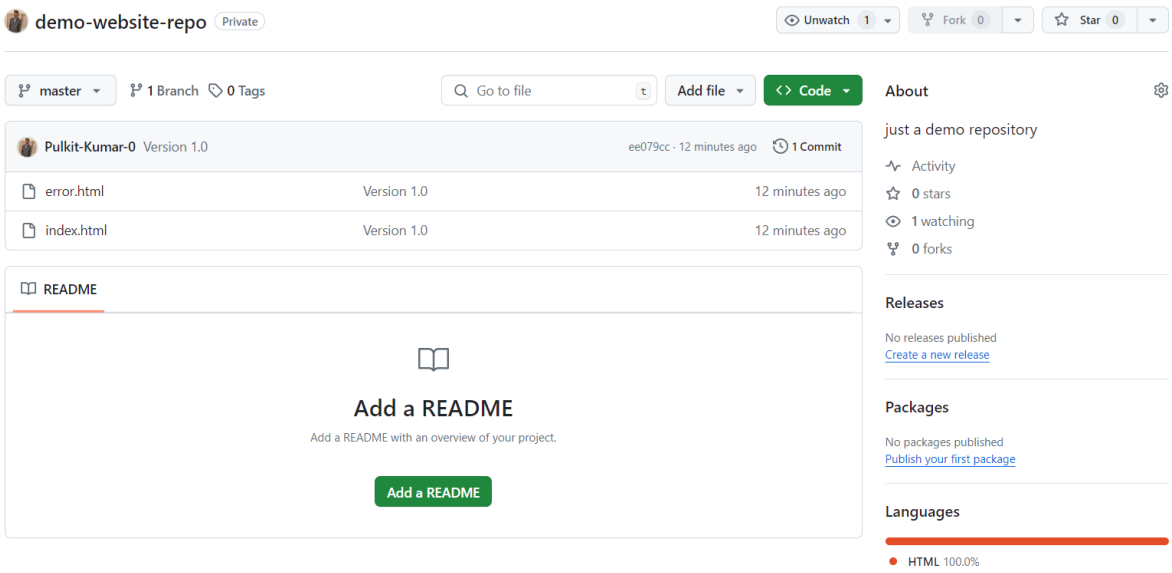

```
PS C:\Users\PULKIT\Downloads\my-website> git remote add origin https://github.com/Pulkit-Kumar-0/demo-website-repo.git
PS C:\Users\PULKIT\Downloads\my-website> git remote -v
origin  https://github.com/Pulkit-Kumar-0/demo-website-repo.git (fetch)
origin  https://github.com/Pulkit-Kumar-0/demo-website-repo.git (push)
PS C:\Users\PULKIT\Downloads\my-website>
```

9. Now we have given the push command to push all our objects to the GitHub repository. Also if you are doing all this for the first time then VS Code will ask you to login first.

   git push origin master

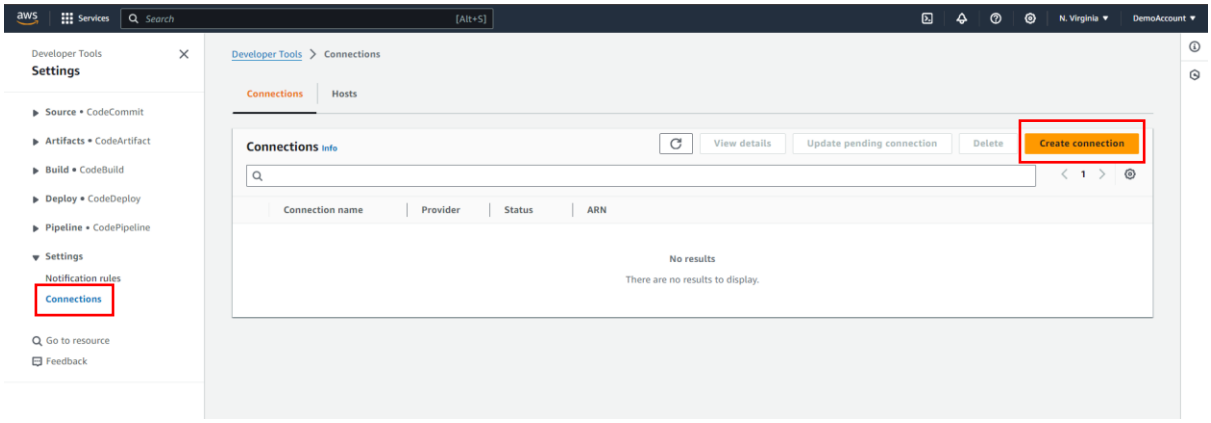
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  AZURE

PS C:\Users\PULKIT\Downloads\my-website> git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 634 bytes | 634.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Pulkit-Kumar-0/demo-website-repo.git
 * [new branch]      master -> master
PS C:\Users\PULKIT\Downloads\my-website>
```

10. Now go to your Repository and click on refresh, you will see your resources there.

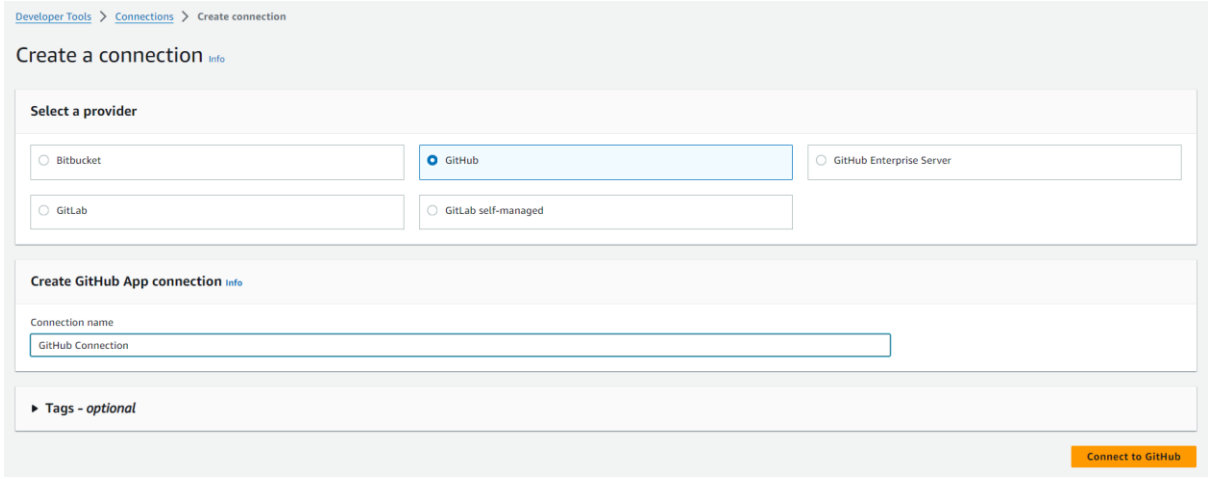


😁 Step 2: Establishing connection between GitHub and AWS

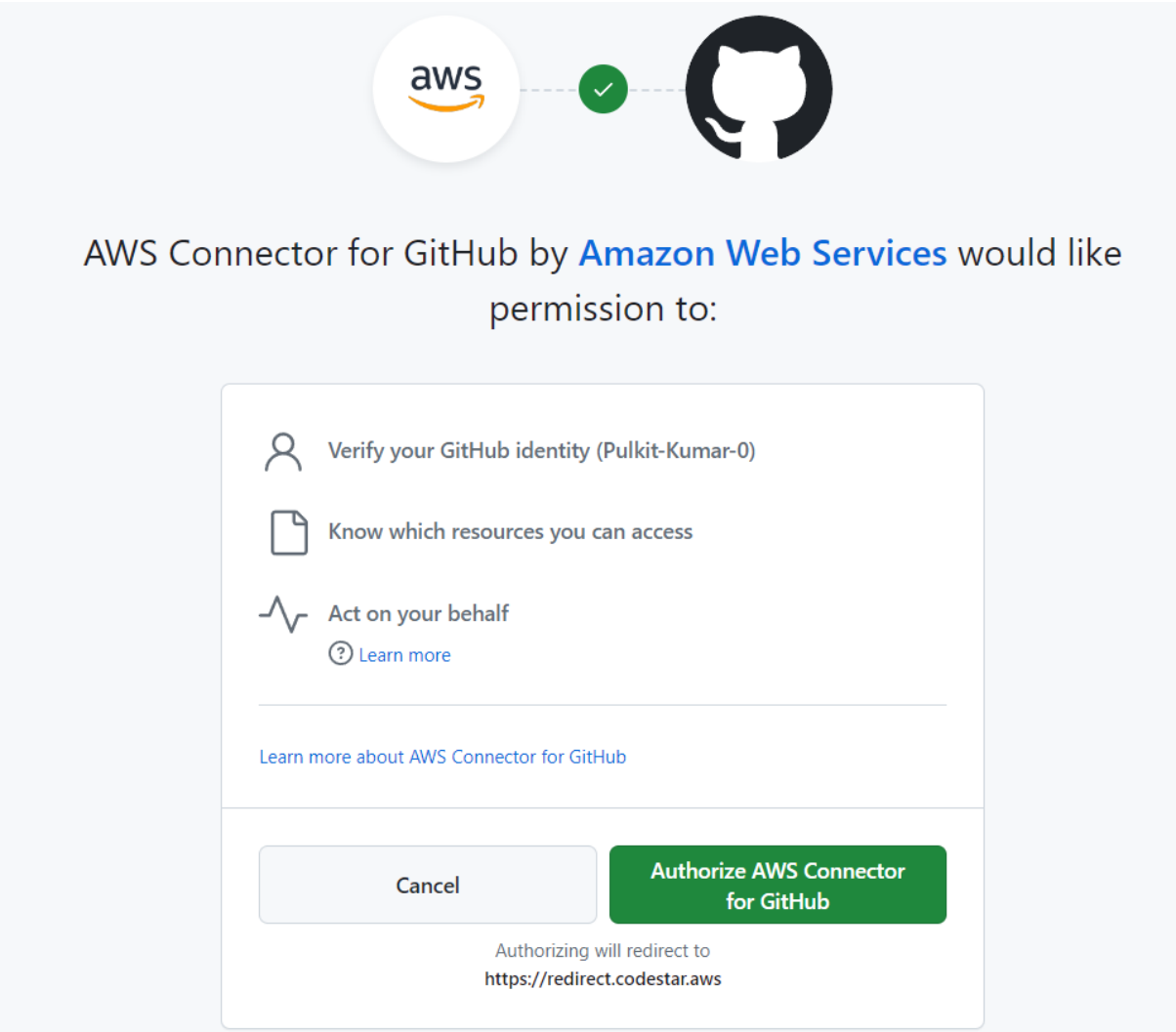
1. Now, before creating a pipeline for Git repositories in any third-party provider, such as GitHub, you must first configure a connection between your AWS account and that provider. So, next, let's go back to the CodePipeline Console to configure the GitHub connection.
2. So, in AWS Developer Tools expand the settings option and choose Connection. Now you need to create a Connection.



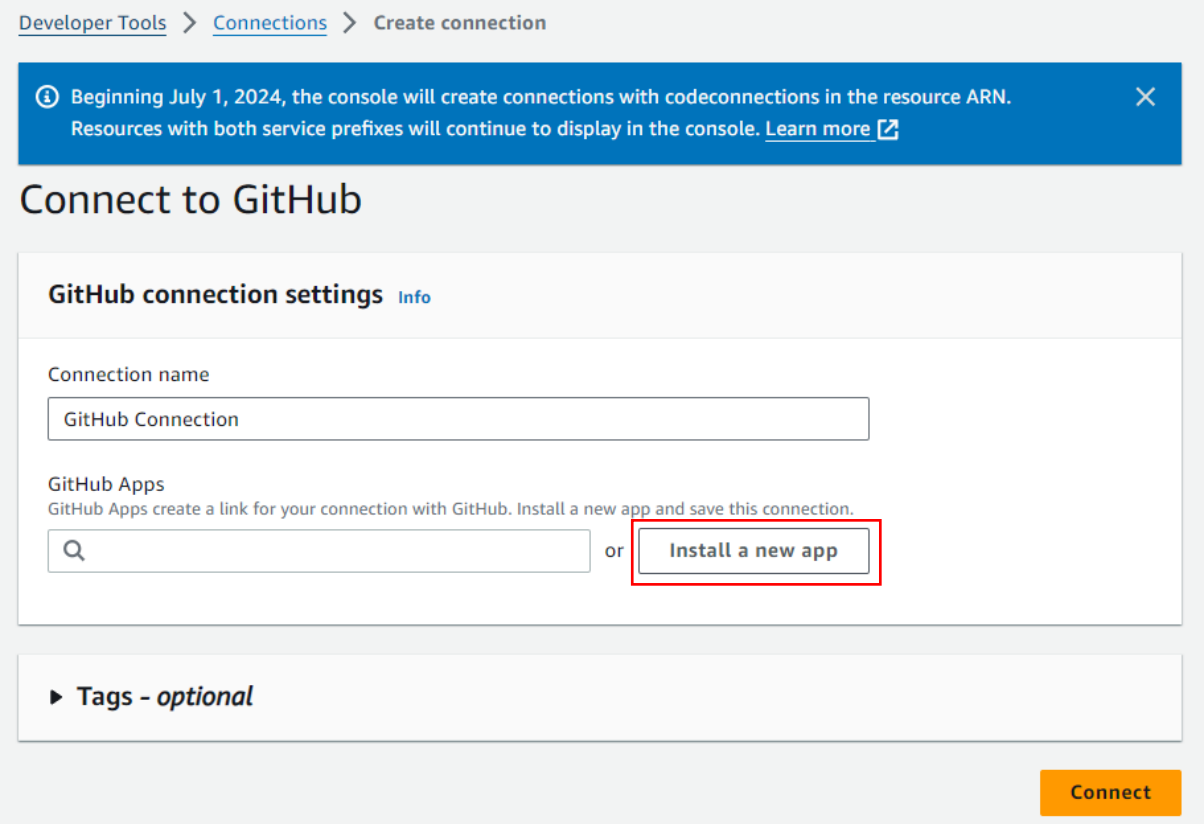
3. Then you need to choose GitHub and give your connection a name, now just click on Connect to GitHub.



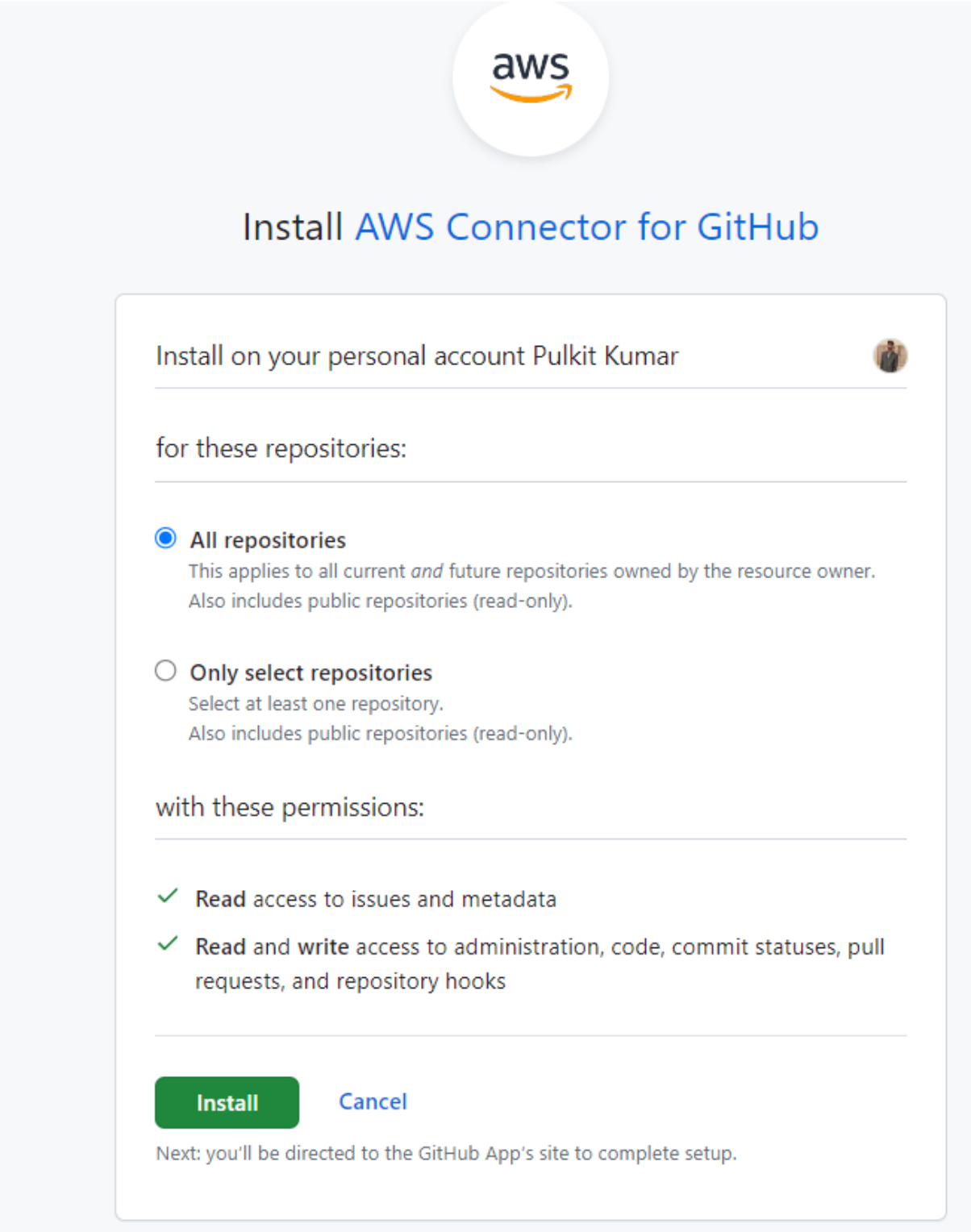
4. As you see, it redirected us to GitHub. This process configures the AWS Connector for the GitHub app. First, you must authorize the AWS connector to access your GitHub account.



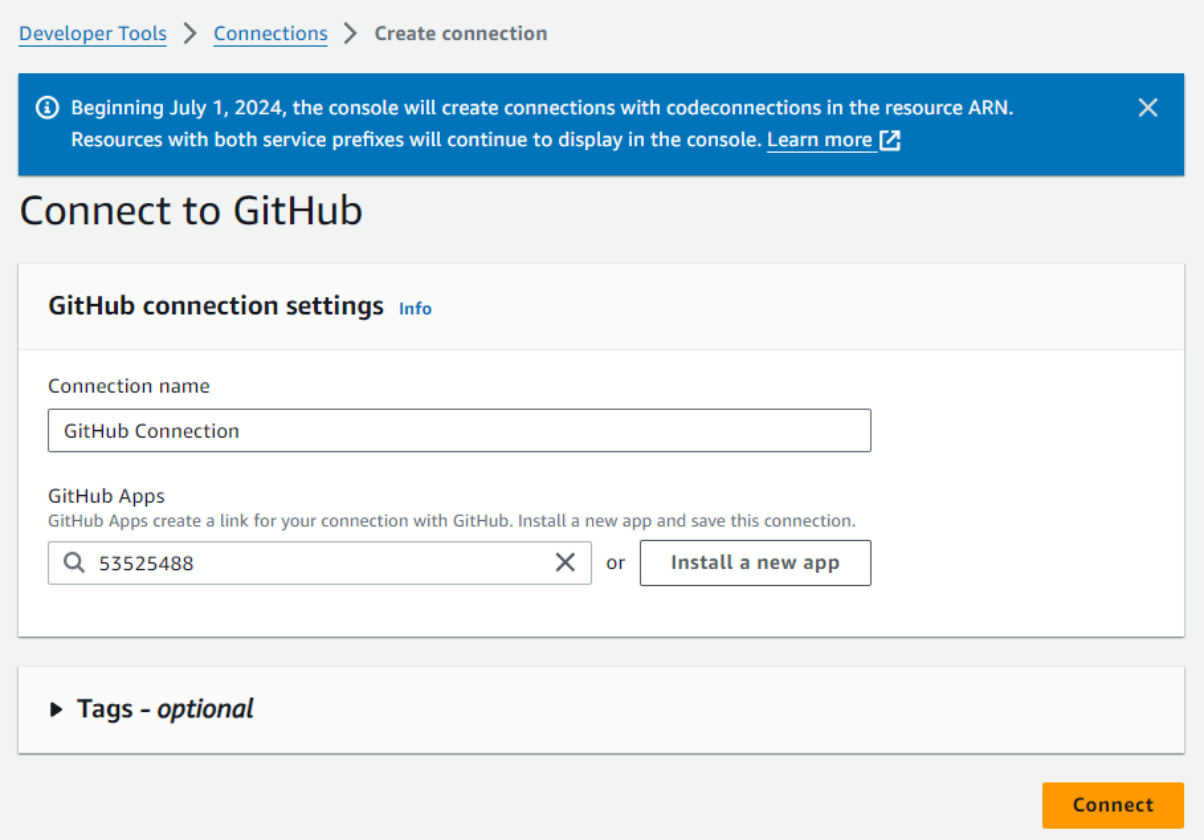
5. Next, you must install the AWS Connector into the GitHub account you will use with your pipelines. If you installed it before, you can choose it from the list here. Otherwise, you can install the app by clicking the button next to it. And, it redirected us to GitHub again.



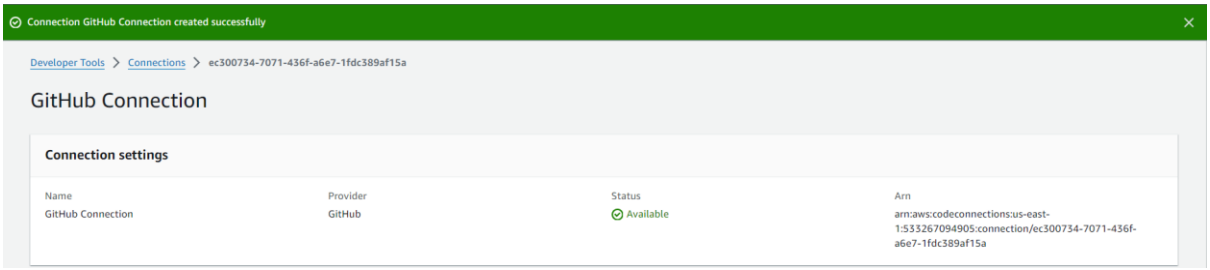
6. Now click on install. After that, it will ask you to enter your GitHub account password.



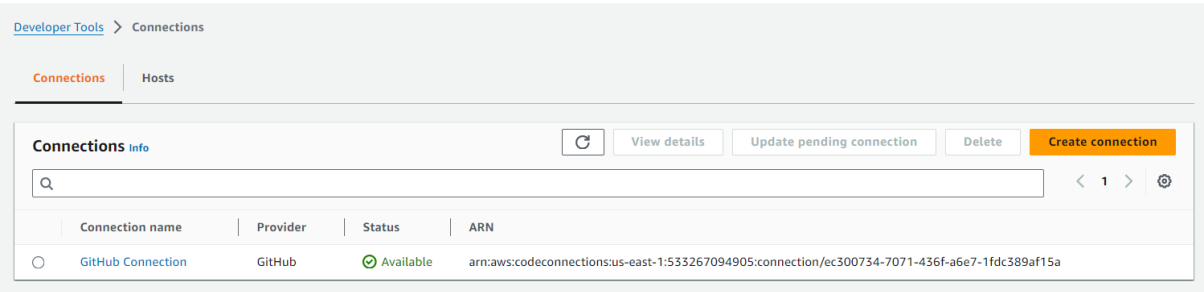
7. The configuration on the GitHub side has been finished. And we are now back to the AWS Developer Tools connections. As you can see, we have a new ID in the 'GitHub Apps' field.



8. Below you can see that our connection was successful. Now, we can use this connection to create a pipeline from the GitHub repositories it has access to.



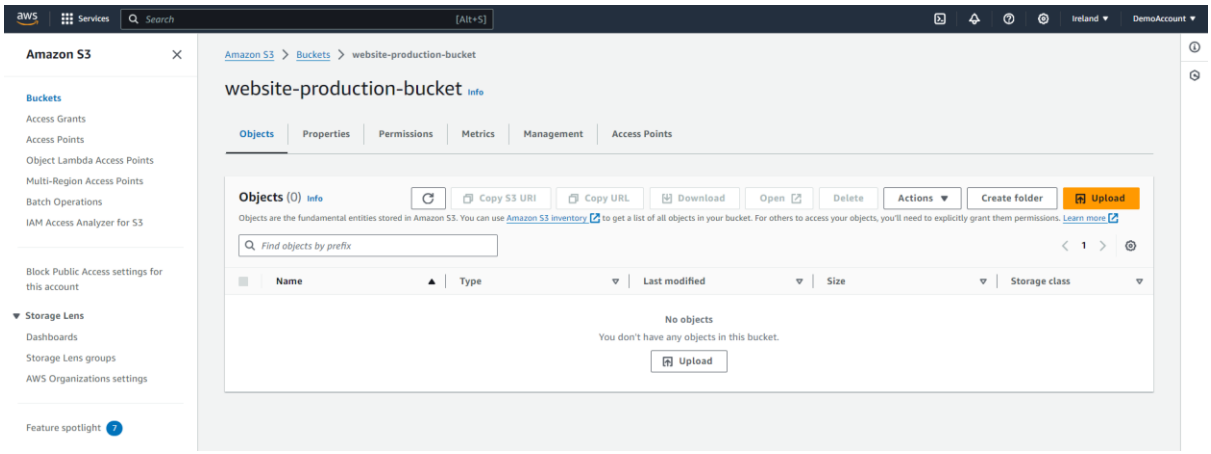
9. Below you can see your connection and the provider is GitHub.



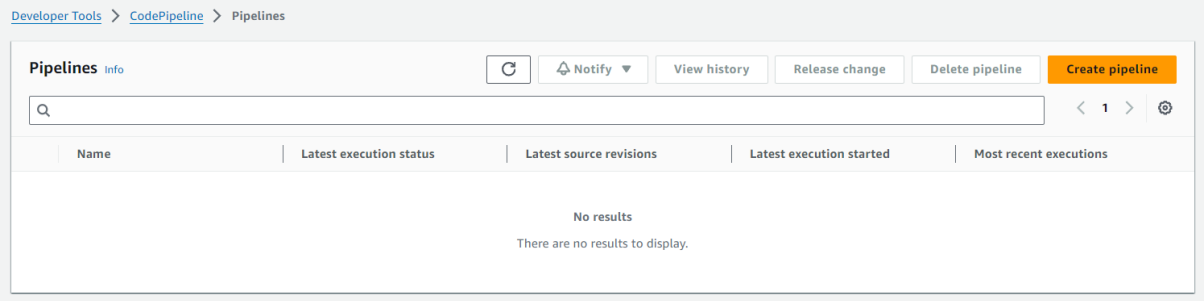
😁 Step 3: Creating Pipeline

1. Now we are going to use the same public bucket that we had used earlier and clear its content. If you don't have a bucket, then you need to create a bucket in the same region

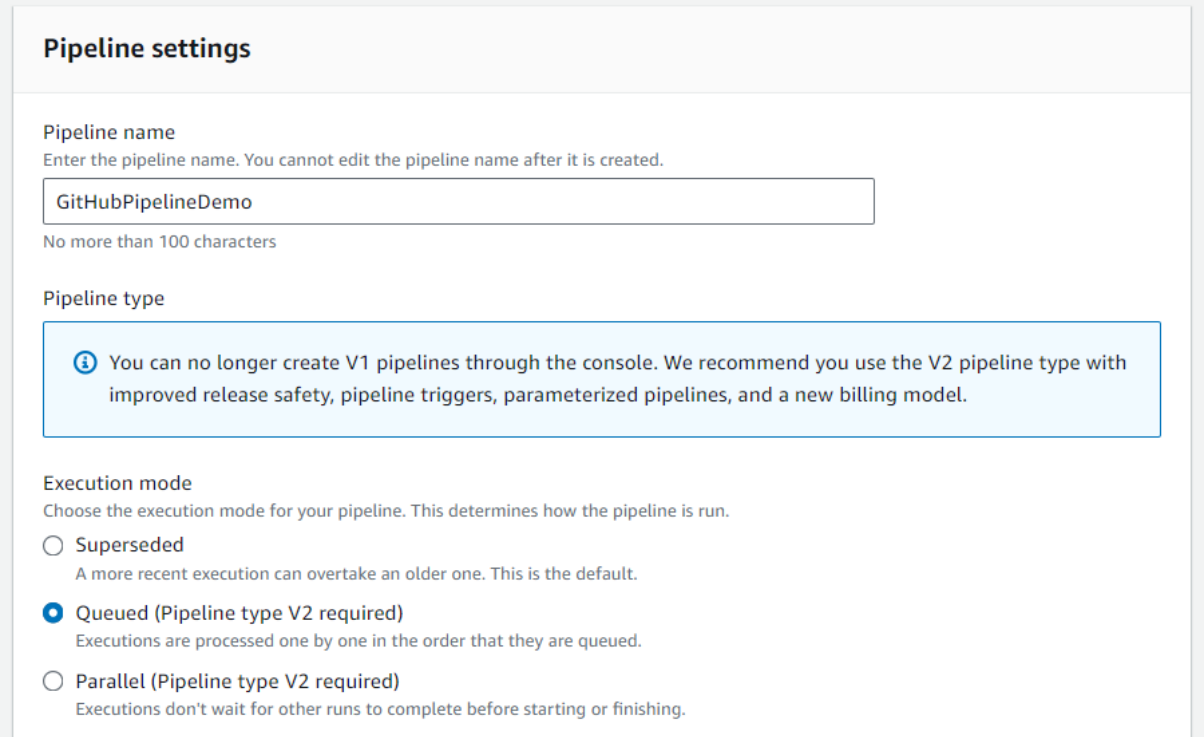
where you will create your pipeline. Make your bucket public update the bucket policy and then enable static website hosting.



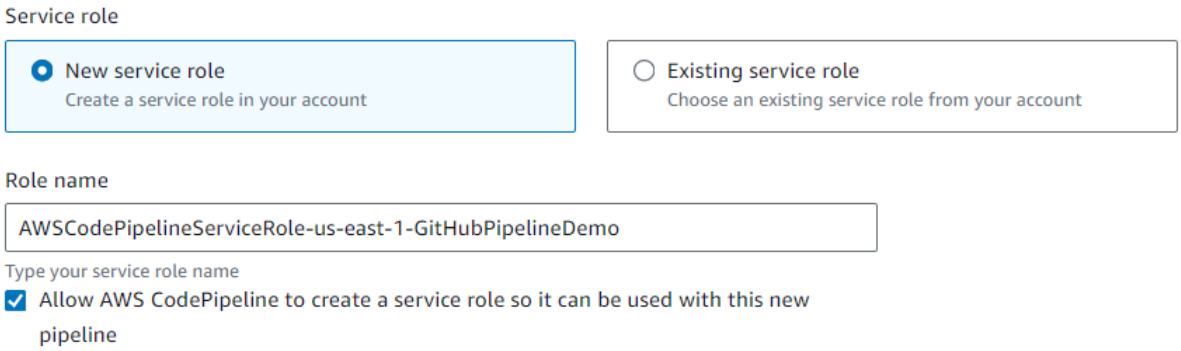
2. So, now go to pipelines and click on Create Pipeline.



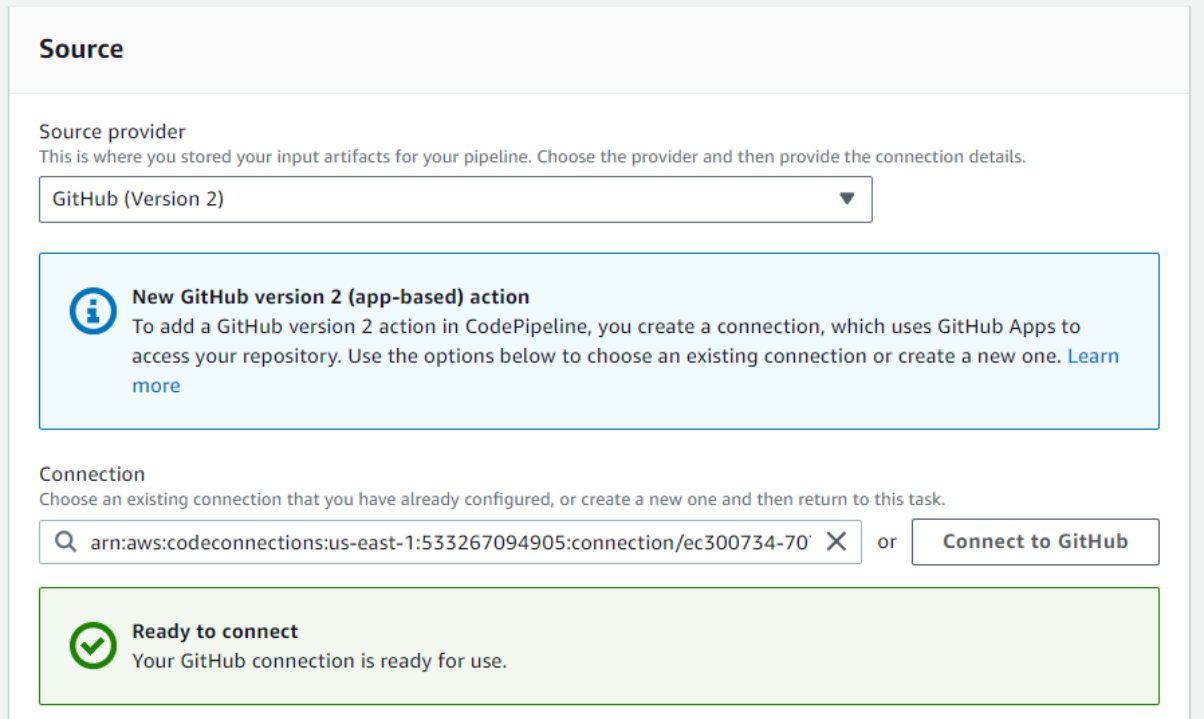
3. Now give your pipeline a name and then keep the rest of the settings to default.



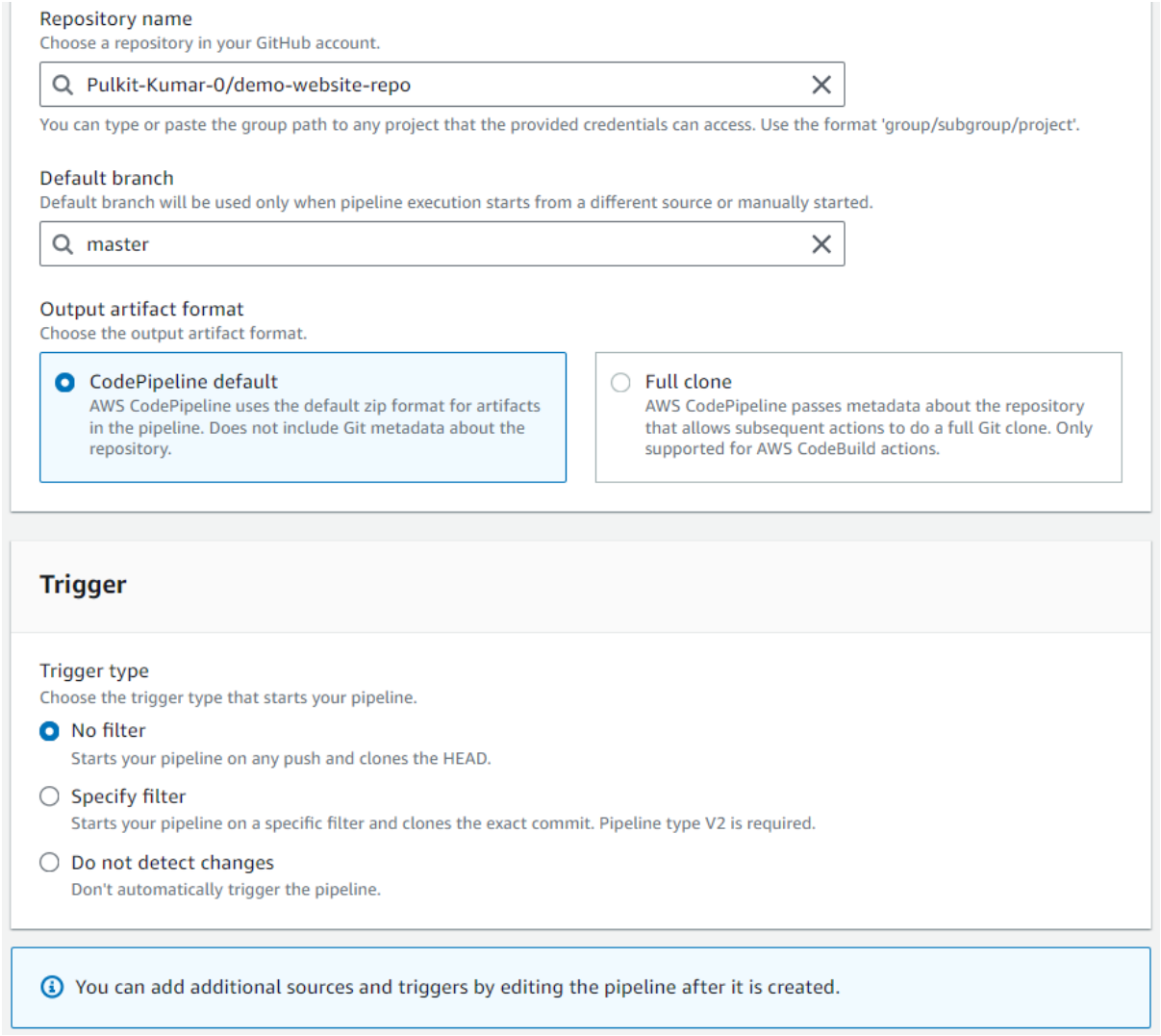
4. Then it will create a service role also. Now move to the next step.



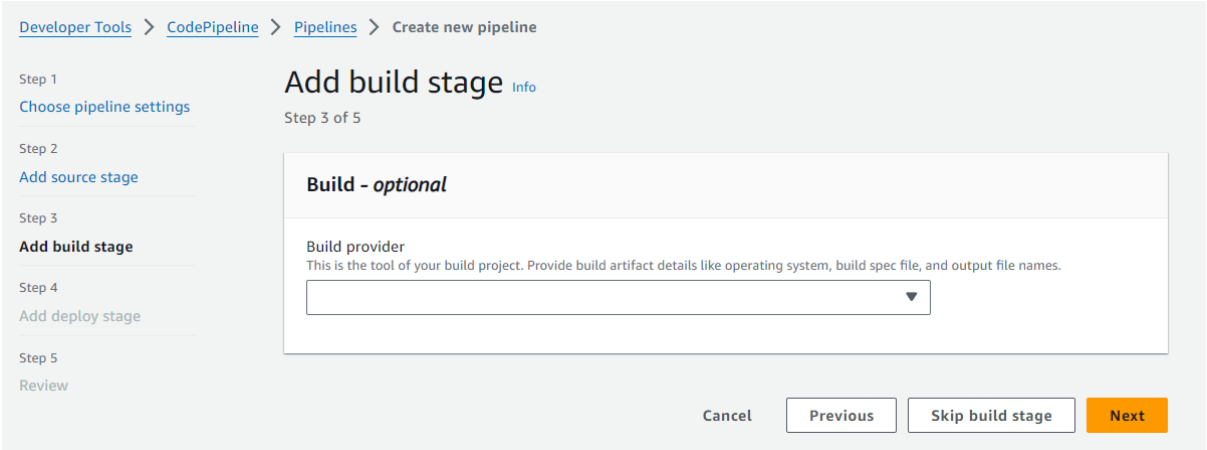
5. In the next step we need to choose our source provider which is GitHub, so choose GitHub (Version 2) and then choose your connection as we have created it earlier.



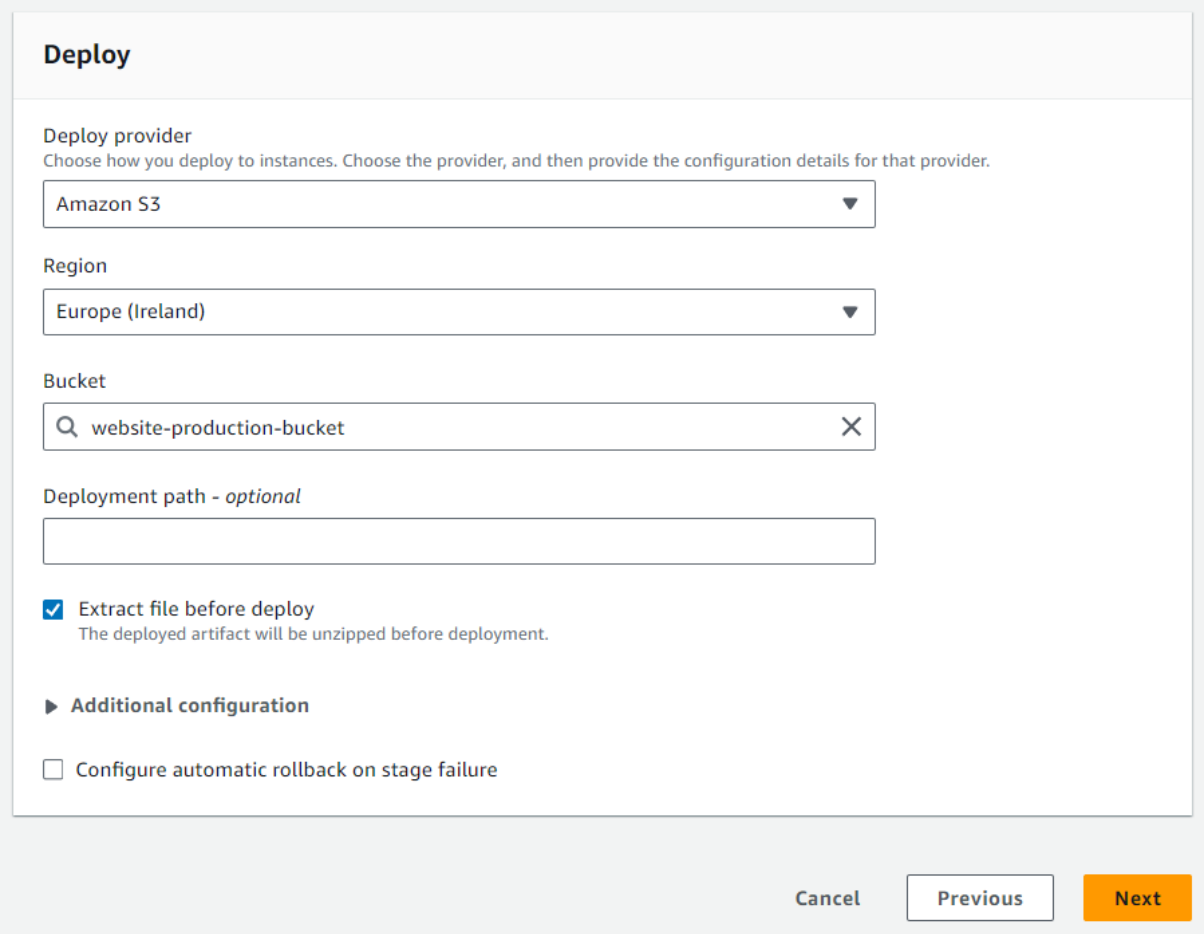
6. After that you need to choose your repository and the default branch. Then in the trigger choose No filter.



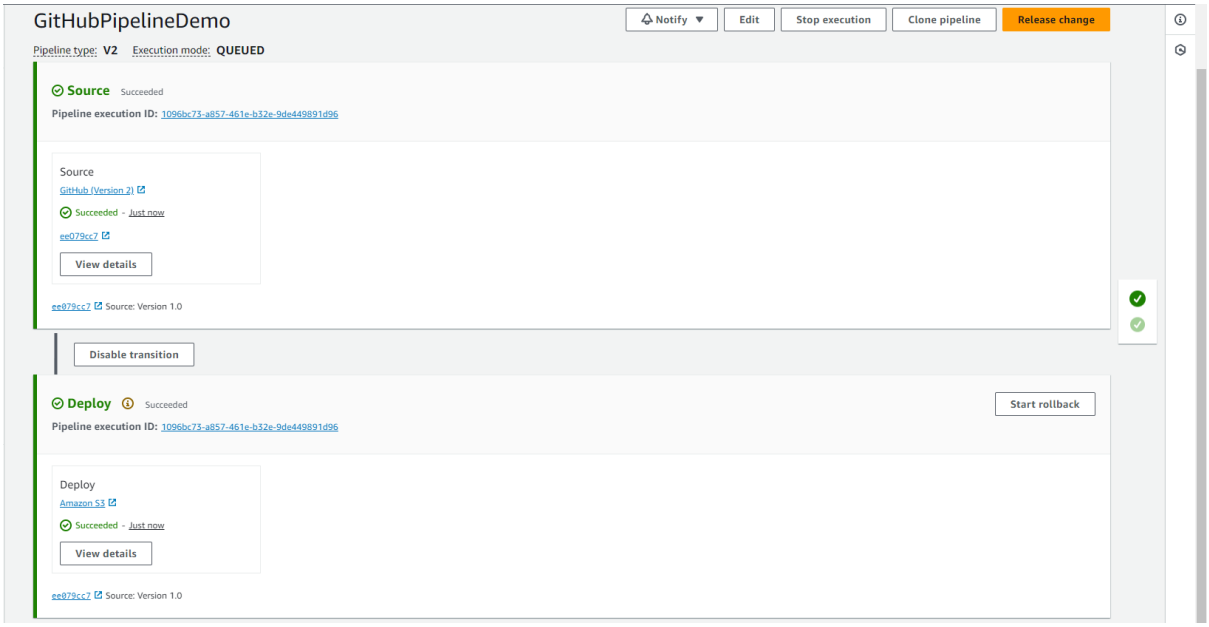
7. Now we will skip the build stage.



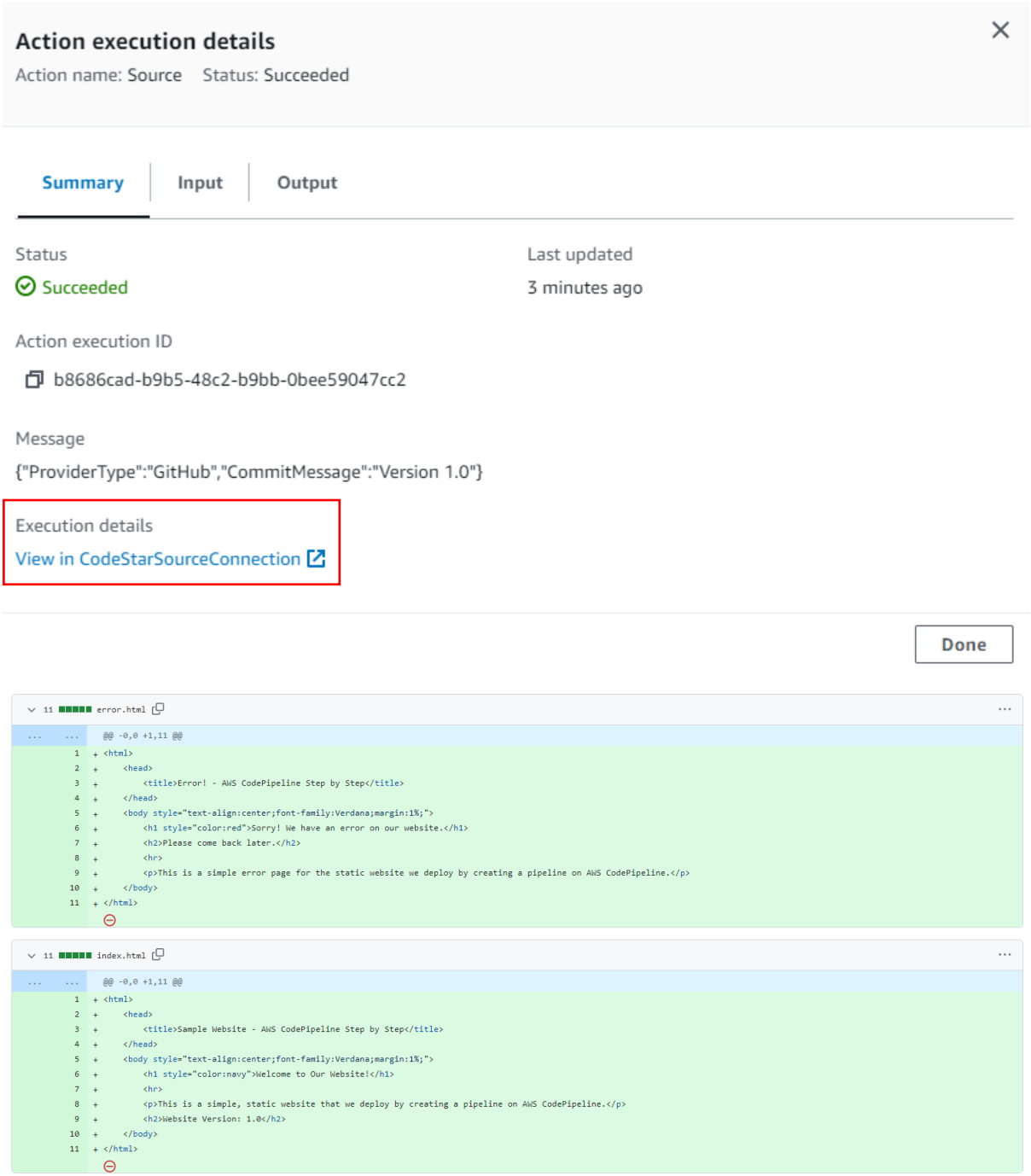
8. In the last step we need to choose where to deploy our content, so choose S3 here and then choose your production bucket. Then move to the review page and create your pipeline.



9. Below you can see that our pipeline has been created and deployed successfully.

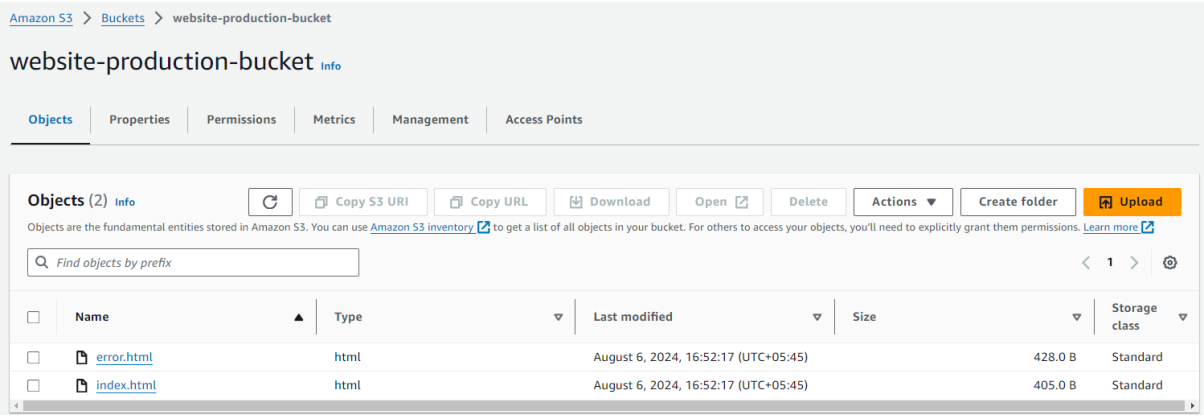


10. Also, if you click on view details from the source then you can see the details regarding your source. Also if you click on view in Code Star Source Connection, then you will be redirected to the commit on your GitHub repository that triggered this action.



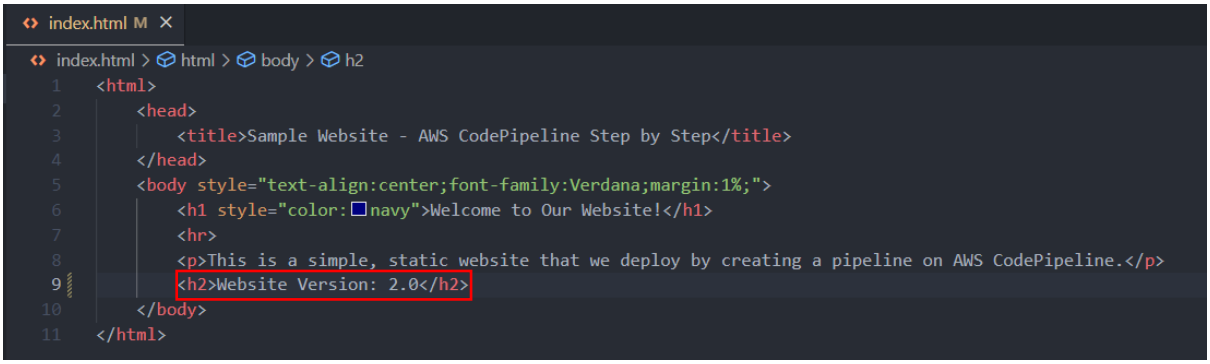
11. Then go to S3 and open your bucket and go to properties, scroll down to bottom click on the static website URL provided and you can see your website.

12. In the objects section you can see your files.



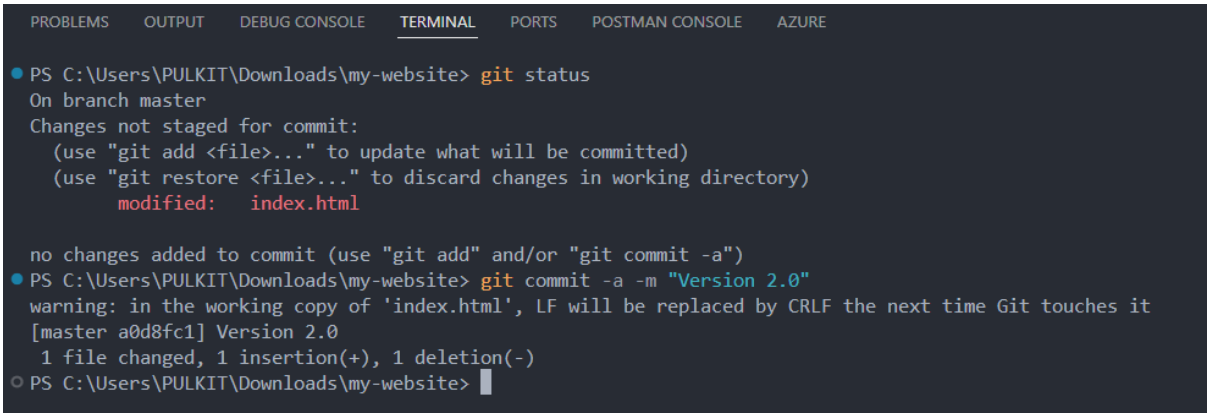
😁 Step 4: Triggering pipeline with GitHub pushes

1. Now go to your local project folder to release a new version as you can see our index.html file and here we just changed the Website version to 2.0 from 1.0 and save our changes.



2. Next, we will use git commands to create a new commit from this change and push it to the master branch of our GitHub repository. And this should trigger our pipeline.
3. In the below snapshot we used two commands first to check the status and we saw that one file needed to be committed, so we issued the commit command.

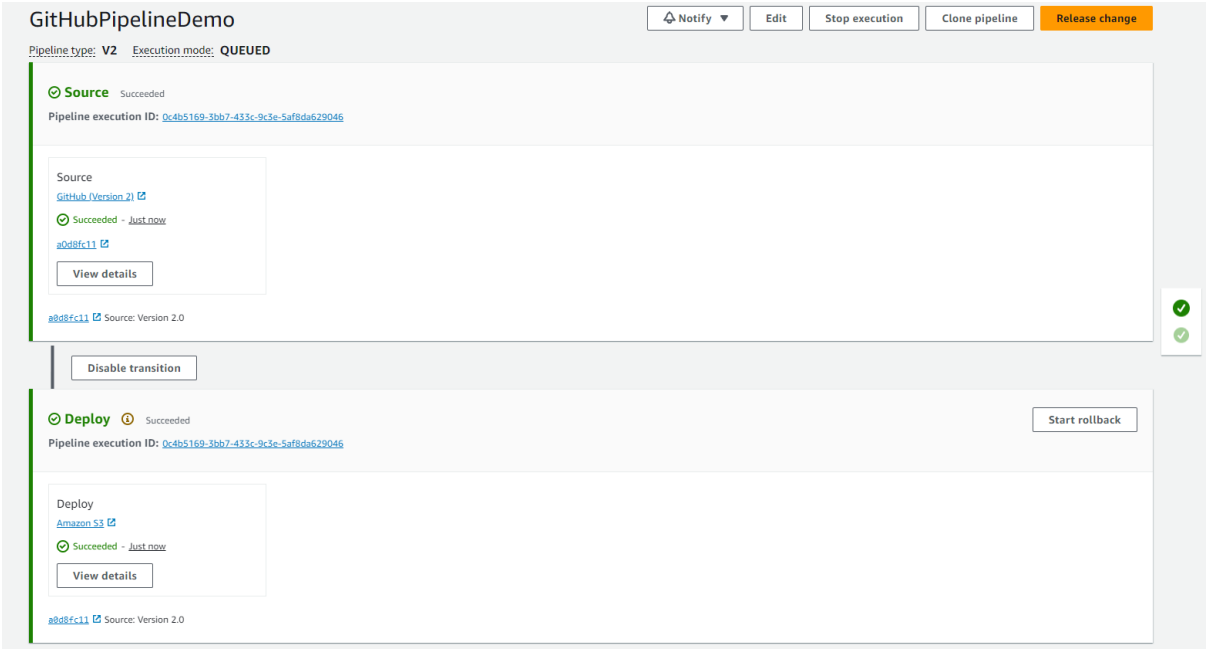
git status
git commit -a -m “Version 2.0”



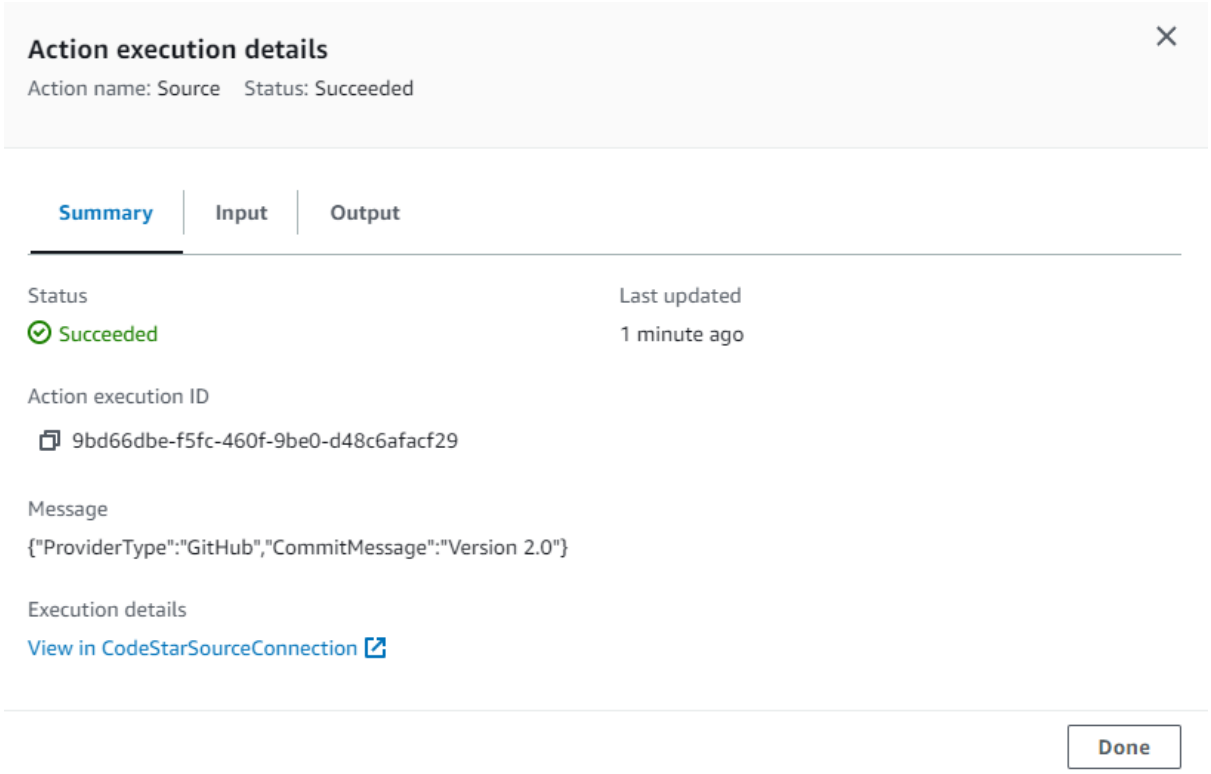
4. Then we first checked the status after that we gave the push command.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  AZURE

PS C:\Users\PULKIT\Downloads\my-website> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\PULKIT\Downloads\my-website> git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Pulkit-Kumar-0/demo-website-repo.git
   ee079cc..a0d8fc1  master -> master
PS C:\Users\PULKIT\Downloads\my-website>
```

5. After that if you go back to your pipeline here you can see that your pipeline just finished running and you can see the version is 2.0.

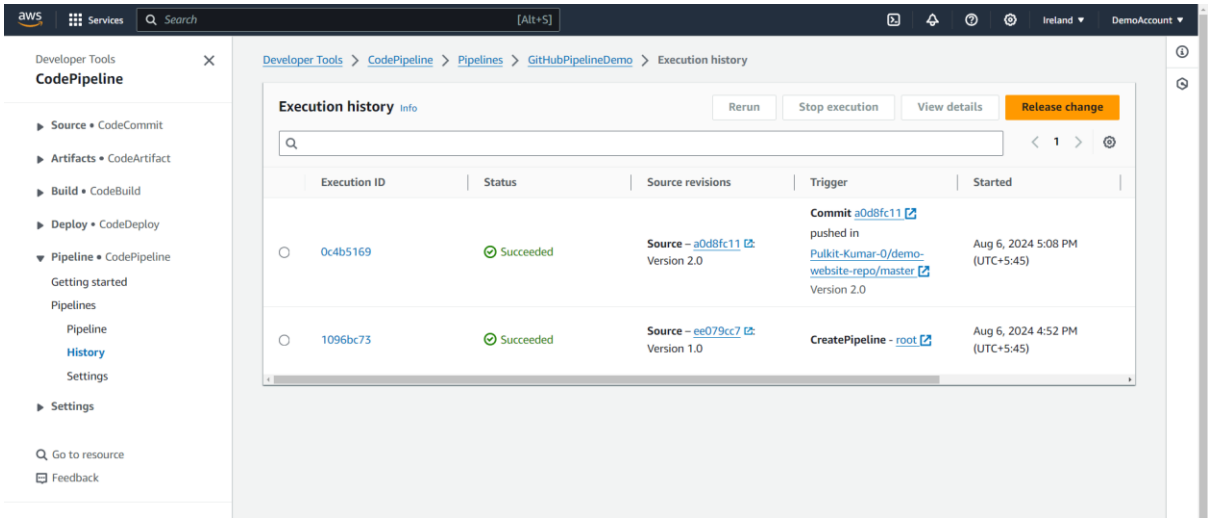


6. Now if you click on view details from the source then you will see the message.



7. Also, if you refresh the website page you will see that the version has been changed to 2.0.

8. Now if you go to the history section from the left pane then you will see the Triggers and the source revisions.


aws
Services
Search
[Alt+S]
Ireland
DemoAccount
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Developer Tools > CodePipeline > Pipelines > GitHubPipelineDemo > Execution history
Execution history Info
Rerun
Stop execution
View details
Release change
Execution ID
Status
Source revisions
Trigger
Started
0c4b5169
Succeeded
Source – a0d8fc11
Version 2.0
Commit a0d8fc11
pushed in
Pulkit-Kumar-0/demo-website-repo/master
Version 2.0
Aug 6, 2024 5:08 PM (UTC+5:45)
1096bc73
Succeeded
Source – ee079cc7
Version 1.0
CreatePipeline - root
Aug 6, 2024 4:52 PM (UTC+5:45)