**MY AWS PROJECTS - MEME MATCHING GAME**

Hi, everyone!

Caleb here!

For this project, I learned how to create a meme matching game using two AWS services; S3 and CodePipeline. GitHub was also used in this exercise.

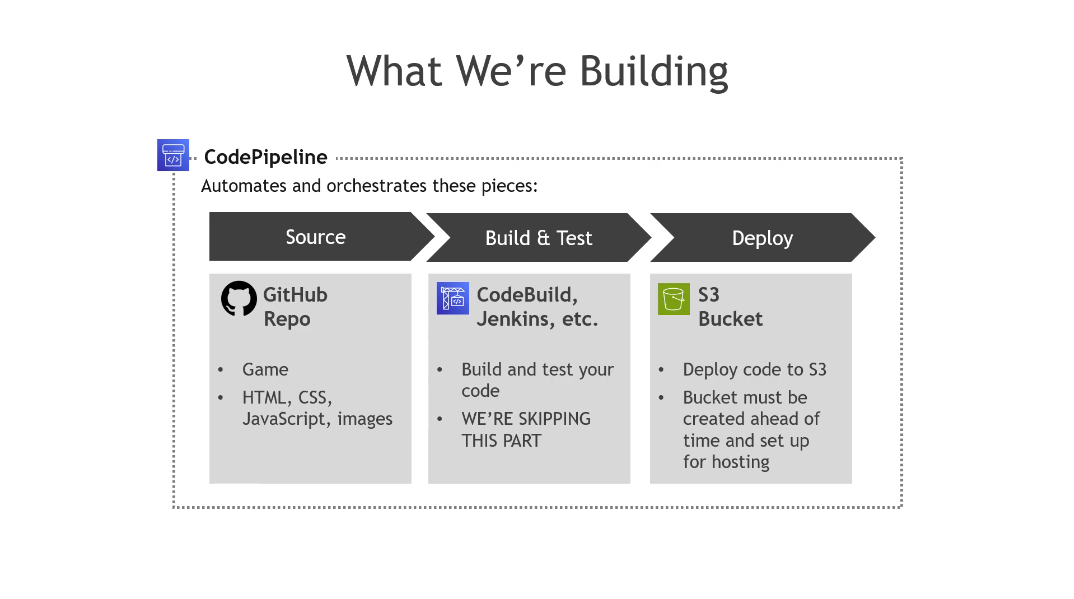
Shouts out to Amber Israelsen for this one, as always. You can find this project on her YouTube channel, <https://www.youtube.com/@TinyTechnicalTutorials>.

Now, let’s get going!

So, this is how the game works: there are ten cards, each of which shows a picture when clicked on, and you are basically clicking on them in pairs simultaneously. When you click on a pair and the pictures are the same, that pair vanishes from the ‘deck’. The whole aim in the end is to clear all the cards.

There is code for this, of course. The idea is to host this code in GitHub, create a pipeline which will pull the code every time a change is made, and then deploy it out to an S3 bucket.

So, here’s a diagram showing the blueprint of what we are attempting for this project:



4.

3

.

2.

1.

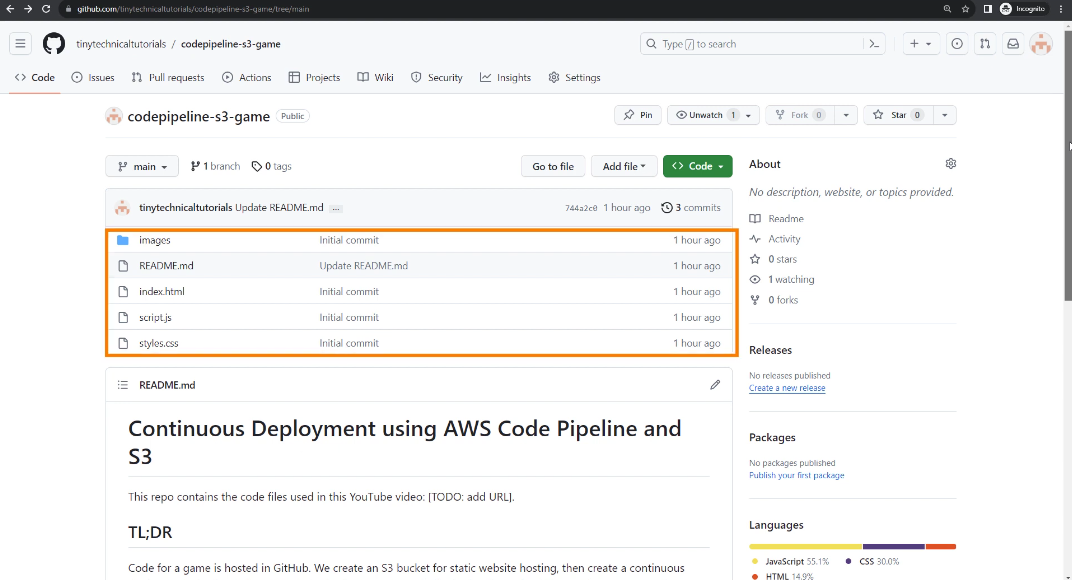
Notice I have numbered the steps for easier clarification.

In step 1, we will be retrieving all the code needed from an existing GitHub repository and placing it in our own GitHub repository.

Step 2 is where there is the building and testing of our code. We will be leaving that step out though, as we already have all the codes needed in their right form and order.

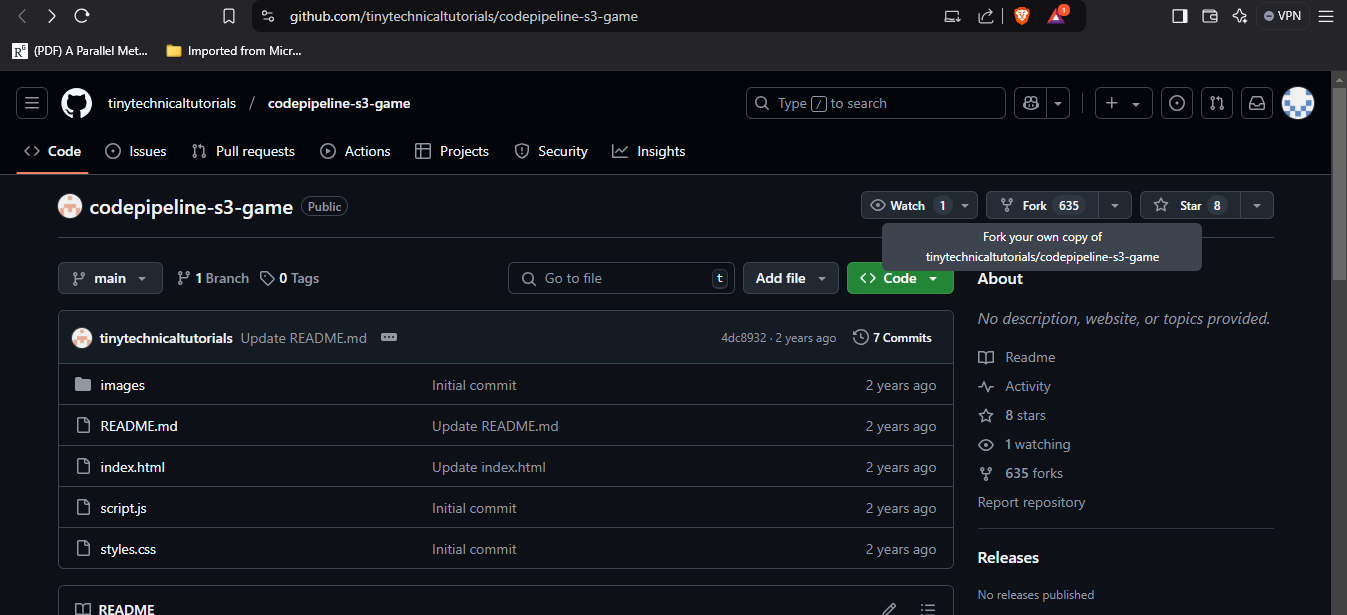
In step 3, we will be creating an S3 bucket in which we will have our static website, where the code will be deployed.

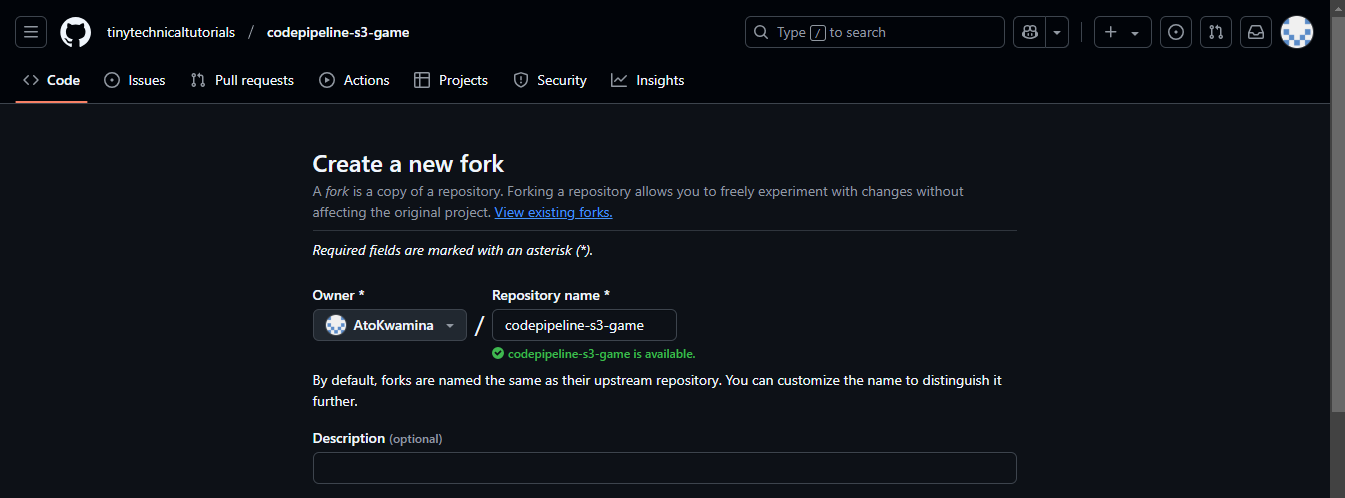
Finally, for step 4, this is where we employ the use of the service known as CodePipeline. CodePipeline will be responsible for automating the process of retrieval of the code from GitHub to S3, as well as automatically updating any changes made to the code and making them take effect in S3.

Let’s start by getting the code needed for this project. We will find these codes at <https://github.com/tinytechnicaltutorials/codepipeline-s3-game>. This repository, once you click on the URL, should look something like this:  


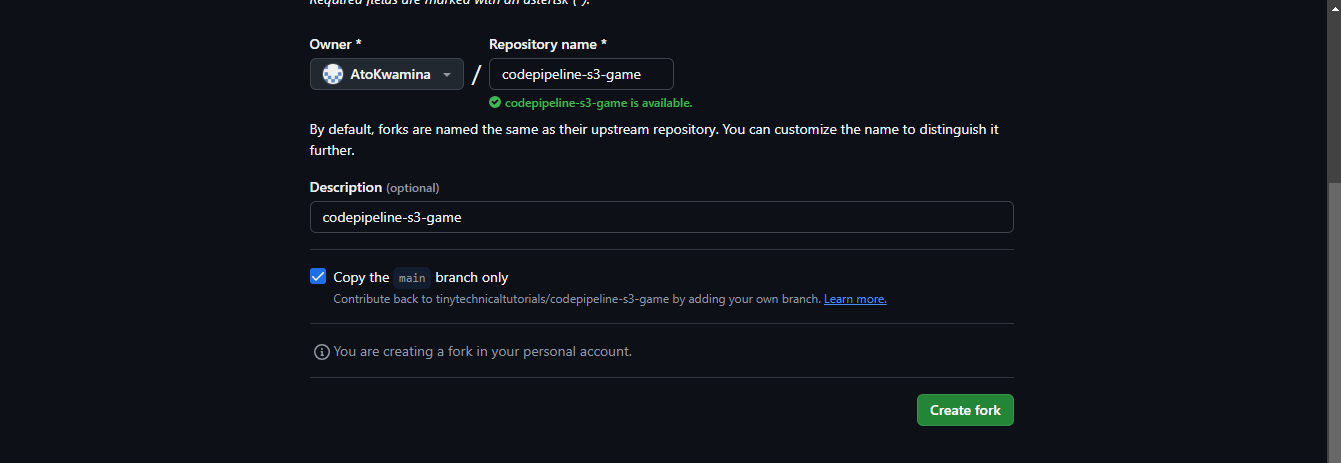
As seen in the highlight in the screenshot, this is where the code needed for the project, as well as a README file giving a heads-up on what is going to be done, are located.

We will need to do something known as ‘forking’; essentially, we are picking up all the codes and sending a copy of it into our own GitHub repository. Of course, that means you will need to have created your own GitHub account to be able to do this. See the diagram below:

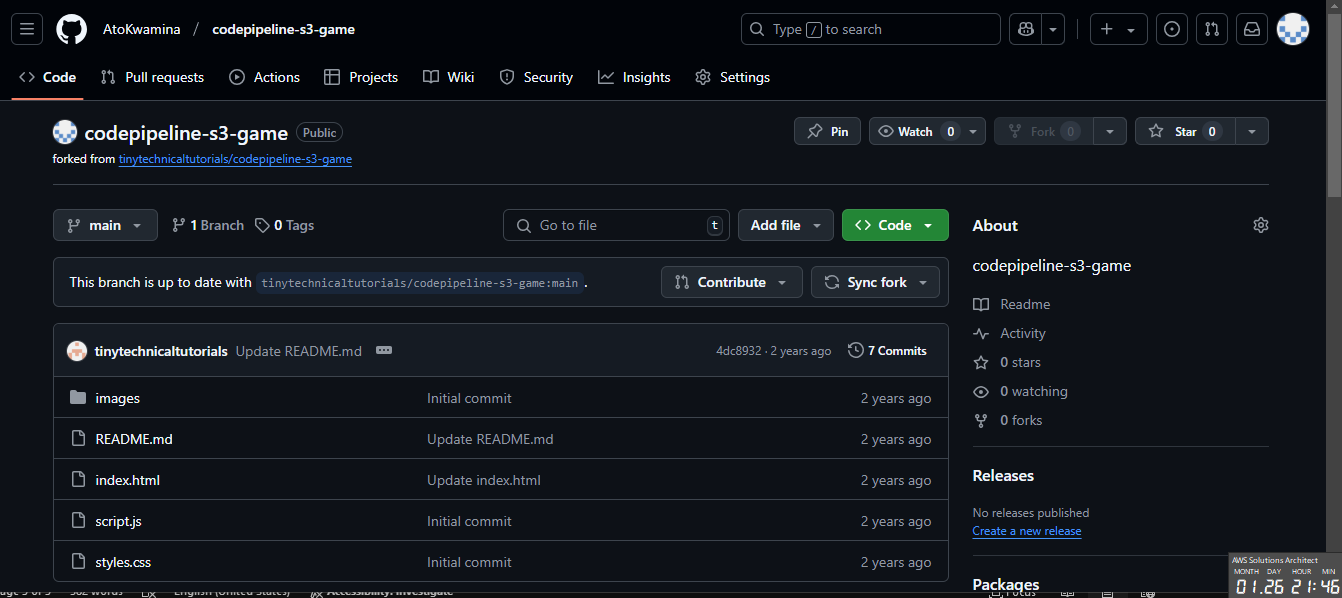


Clicking on the ‘fork’ option should show the following page:  


You may choose to leave everything as-is, and then hit ‘Create fork’ to mirror the contents in your own repository. For the ‘Description’ part, I just decided to give it the same as the name of the repository. Feel free to add any description you want.

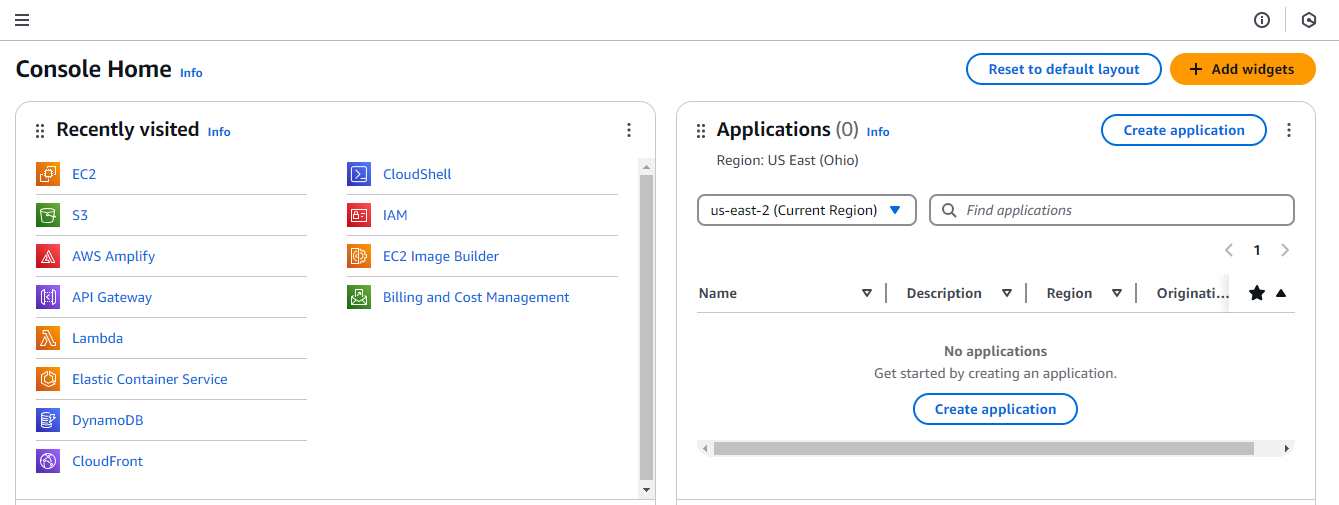


As now shown, the repository has been created in my own account ‘AtoKwamina’.

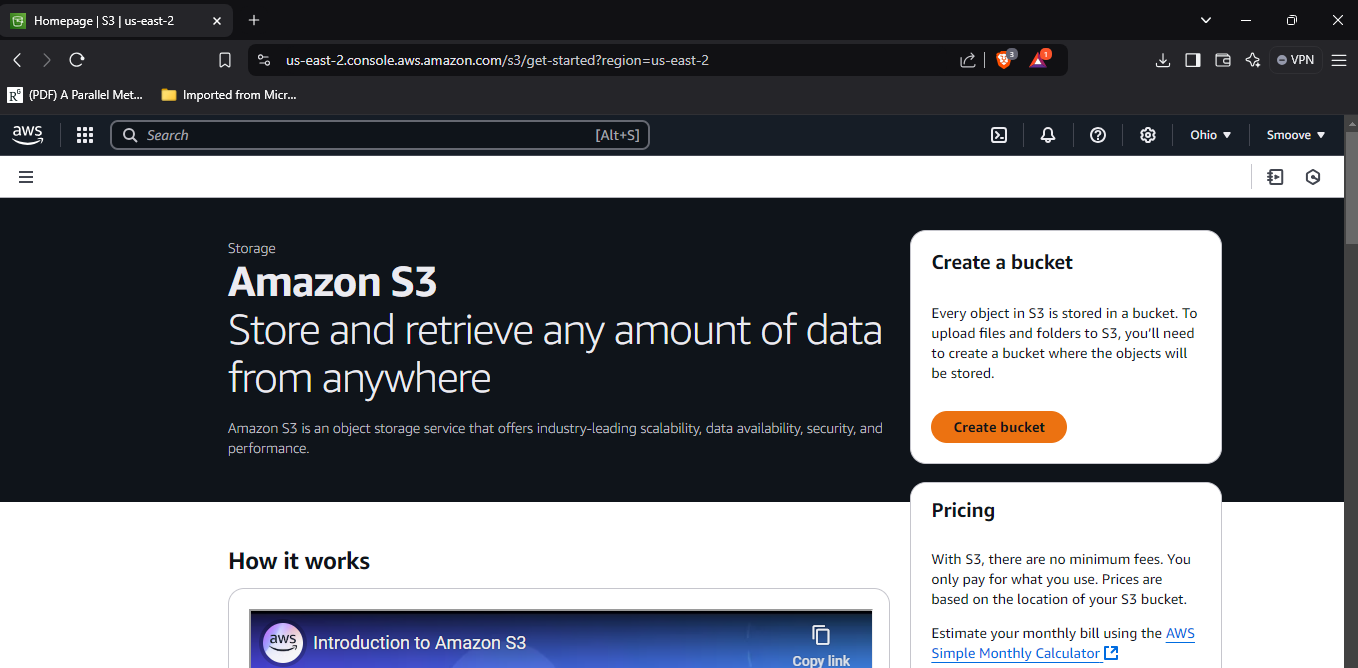


Next, we will need to create and configure an S3 bucket. S3 is used for ‘object’ (file) storage on AWS. It can also be used for static website hosting, which is pretty much what we are trying to do at the moment.

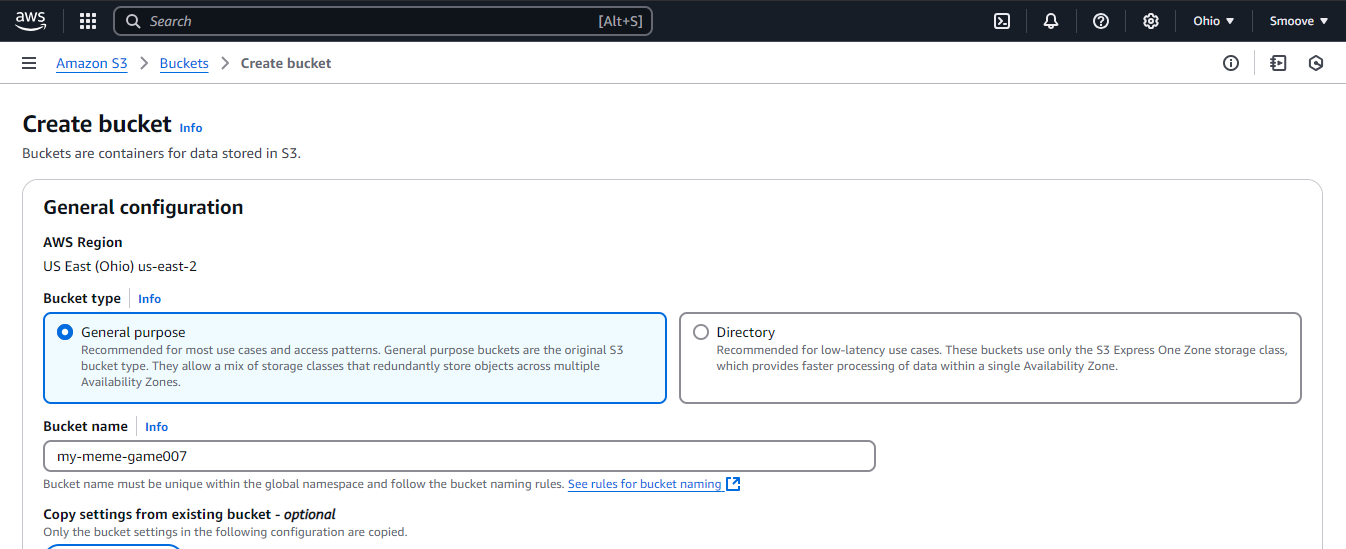
So, let’s move now to our AWS console and navigate to S3:



We then go ahead and begin the bucket creation process:

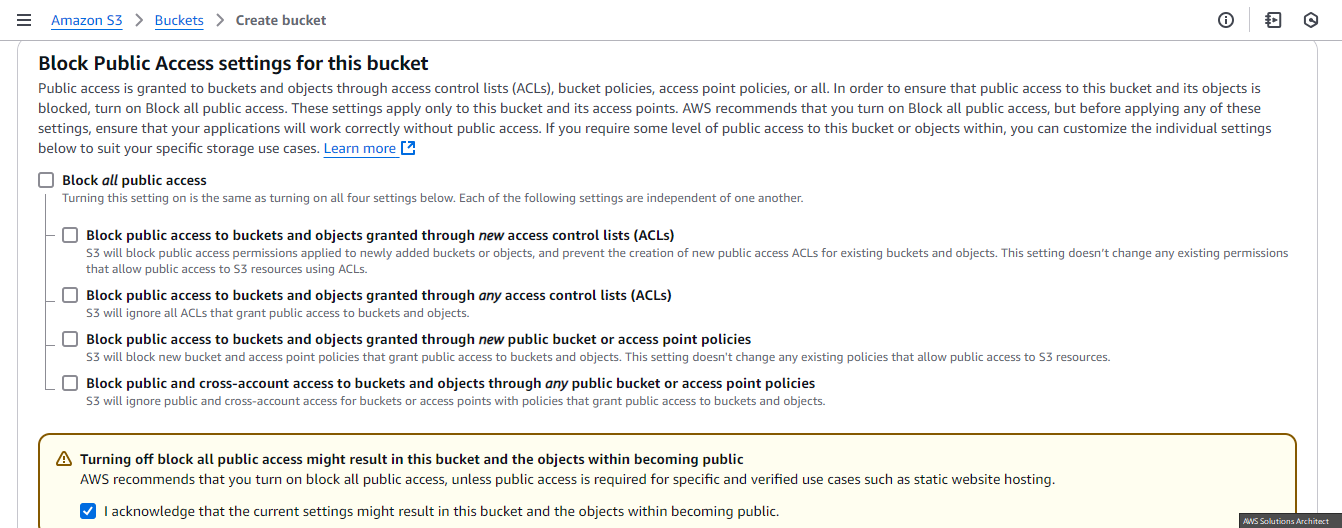


First thing we do is give it a name. I just decided to use the name my instructor, Amber, used. Make sure to add something to it though, if you decide to do the same; remember S3 bucket names are globally unique, so if you write it without any additions, there is a chance it could already be in use.

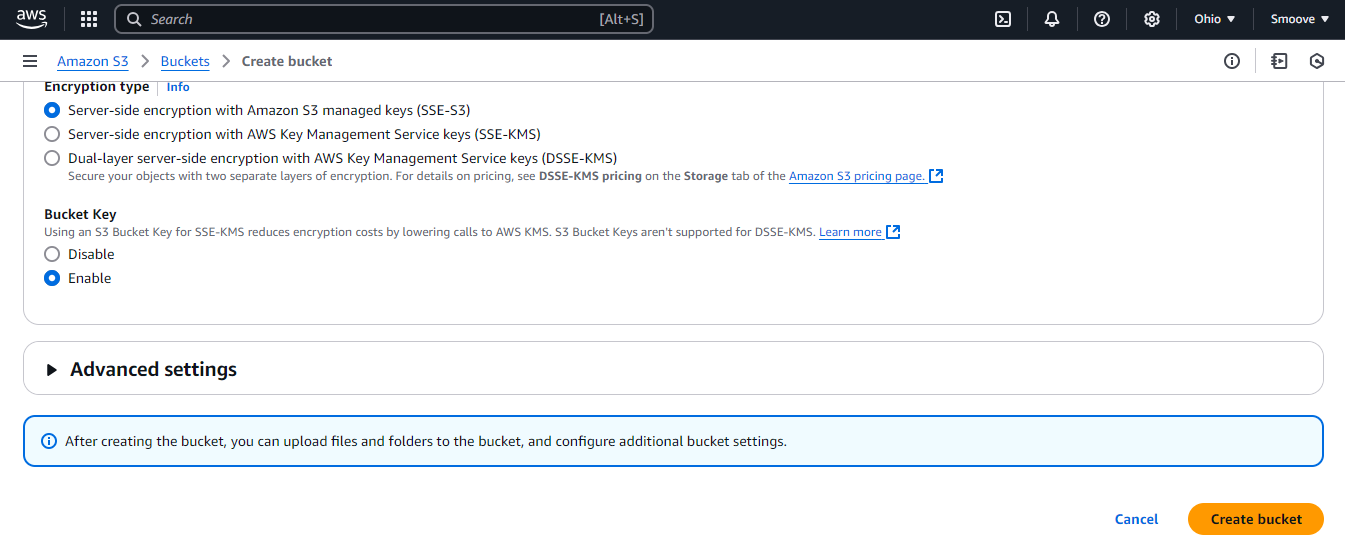


Scroll down till you get to the section with the inscription ‘Block *all* public access’ and deselect that option. We are doing this because we are creating a static website that we want to be globally accessible. In a real-world environment though, be careful about that option.

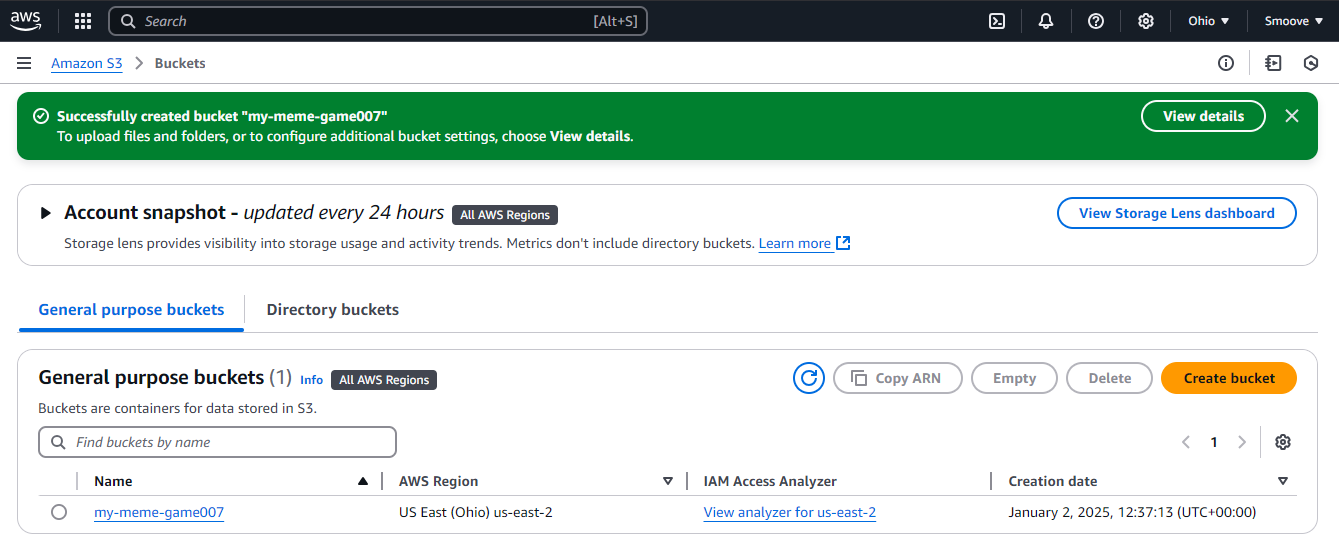
Also tick the acknowledgment part to let AWS know you are sure of what you are doing.



Now scroll down and leave everything else as-is then click on ‘Create bucket’.



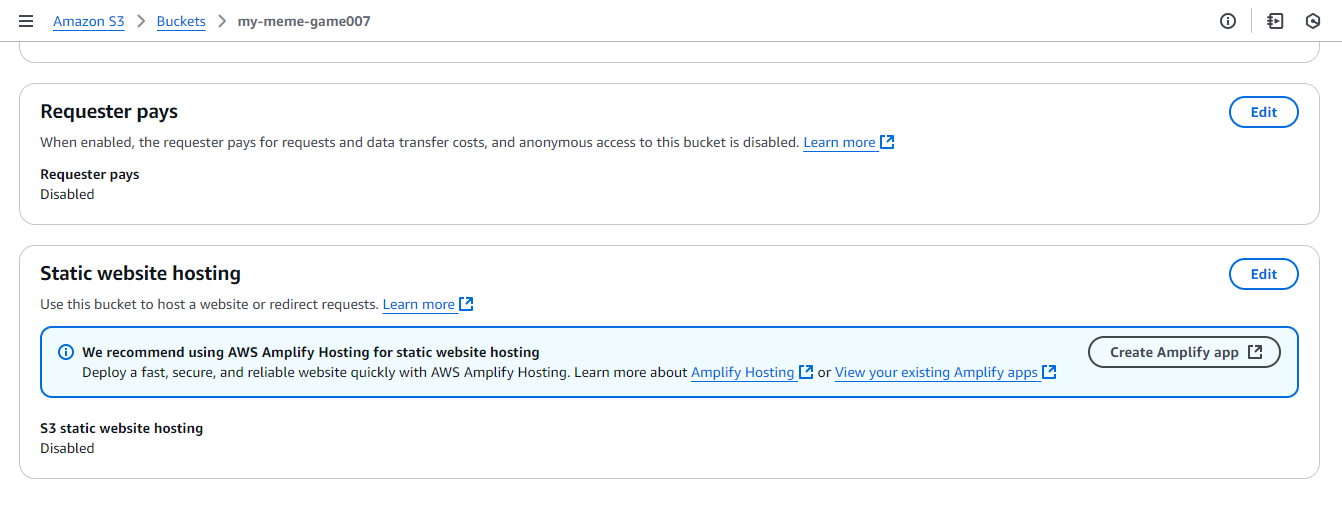
Now you can click on ‘View Details’, or just select the bucket just created by clicking on it under ‘General purpose buckets’.



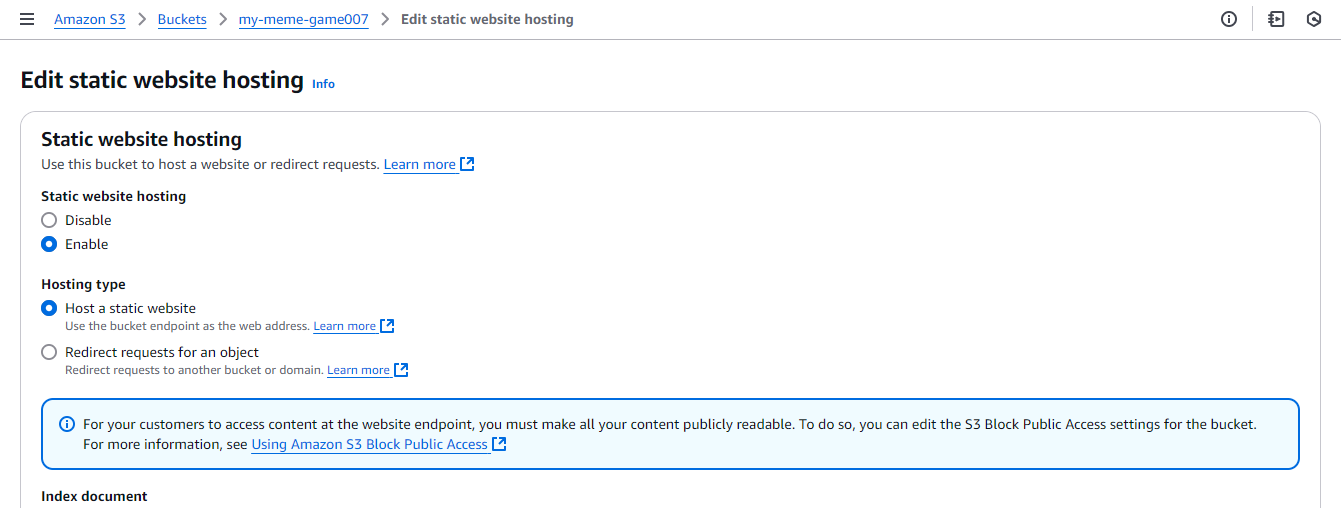
There are still a few updates we will need to configure for this bucket, so on this next page, go to ‘Properties’.



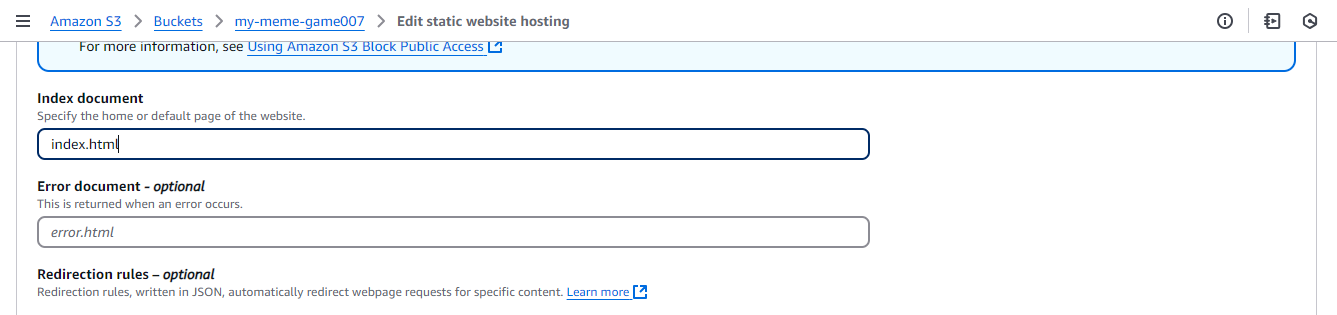
First thing we will need to do here would be to enable static website hosting, as it is disabled by default. So, we will scroll all the way down on this page to where we see that option, and then click on ‘Edit’.



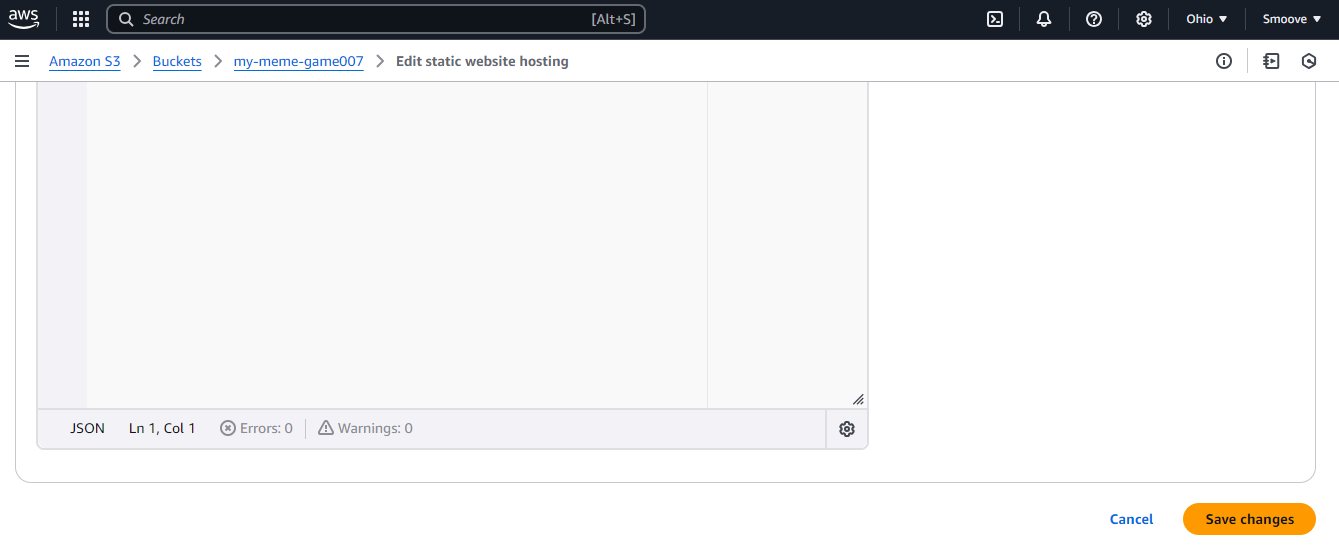
Click on the ‘Enable’ option to enable it.



Our next change will be for the index document, which will be our default homepage of ‘index.html’. Type that in there.

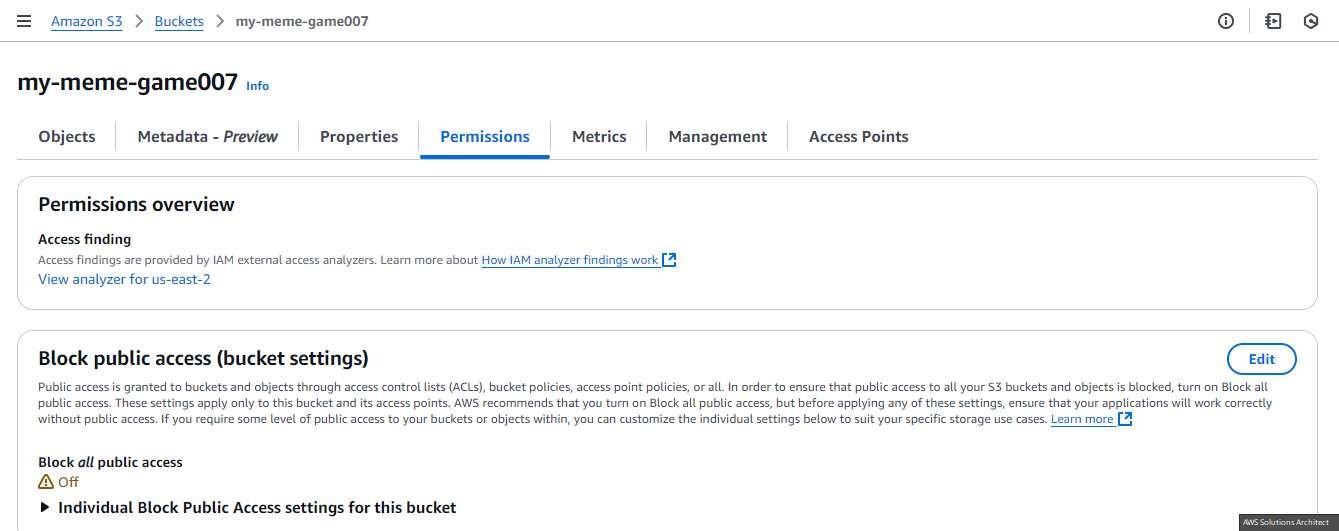


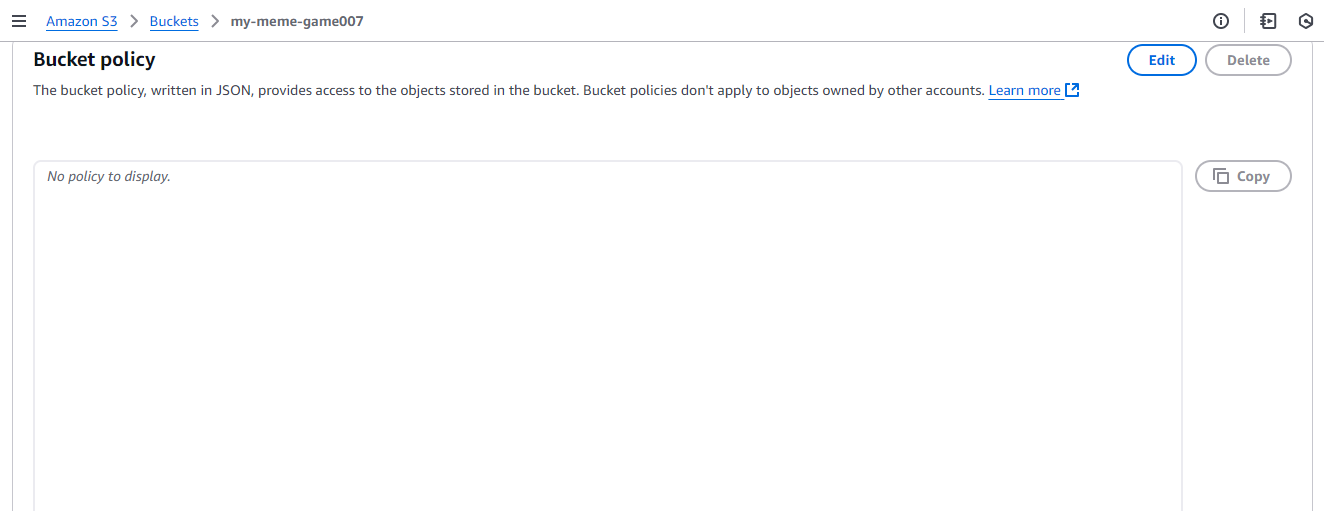
Then scroll down and save changes.



Next thing we need to look at now is permissions. We had previously disabled the ‘Block all access’ setting, but to add to that, we would also want global access to our website to view our files. This is where the bucket policy setting comes in handy.

Go to ‘Permissions’, scroll down to the ‘Bucket policy’ section and click on ‘Edit’.





You’ll need this JSON code to edit the bucket policy to allow read access to the files in your bucket.  
Don’t forget to update the bucket name with the name of your particular bucket.

{

    "Version": "2012-10-17",

    "Statement": [

     {

         "Sid": "PublicReadGetObject",

         "Effect": "Allow",

         "Principal": "\*",

         "Action": [

             "s3:GetObject"

         ],

         "Resource": [

                "arn:aws:s3:::Bucket-Name/\*"

         ]

     }

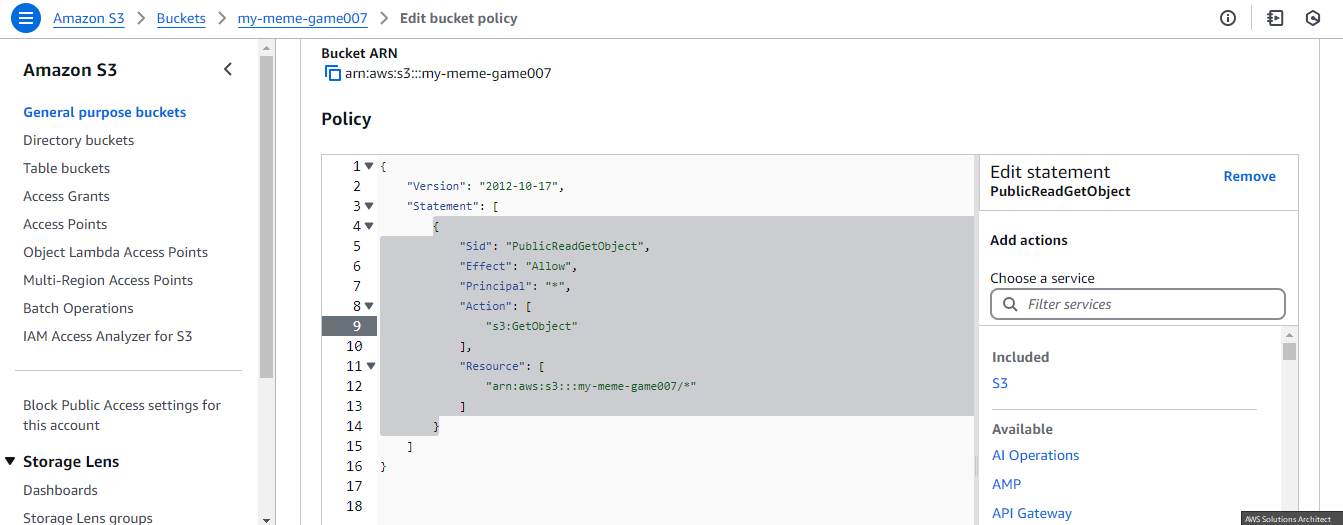
    ]

}

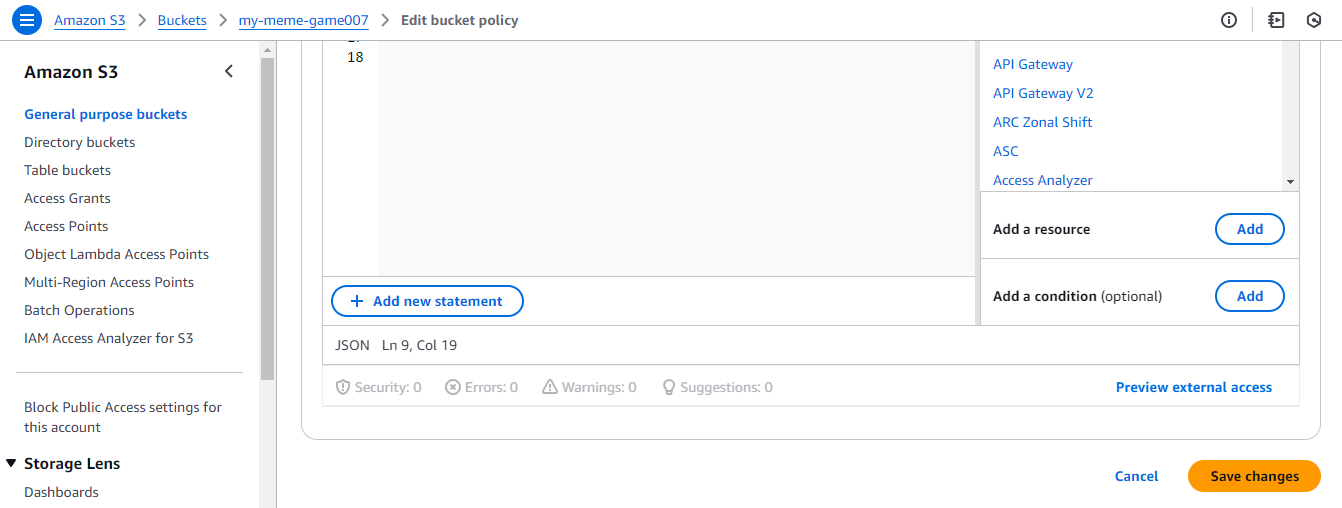
Let me try and break down the code a little for easier understanding.

Essentially, the Sid part is saying ‘public read access to the bucket’, Effect says ‘allow’, meaning that ‘allow this access’, the ‘\*’ for Principal means ‘everyone’, the Action part is ‘getobject’, meaning ‘read’, that is, everyone may access and view the contents of the bucket, and then finally, the ‘arn’ information is the ‘Amazon Resource Name’, which in this case is the name of our S3 bucket. That’s why the name of our bucket was needed there.

As you’ll notice in the screenshot below, I have updated the policy and used the name of my S3 bucket.

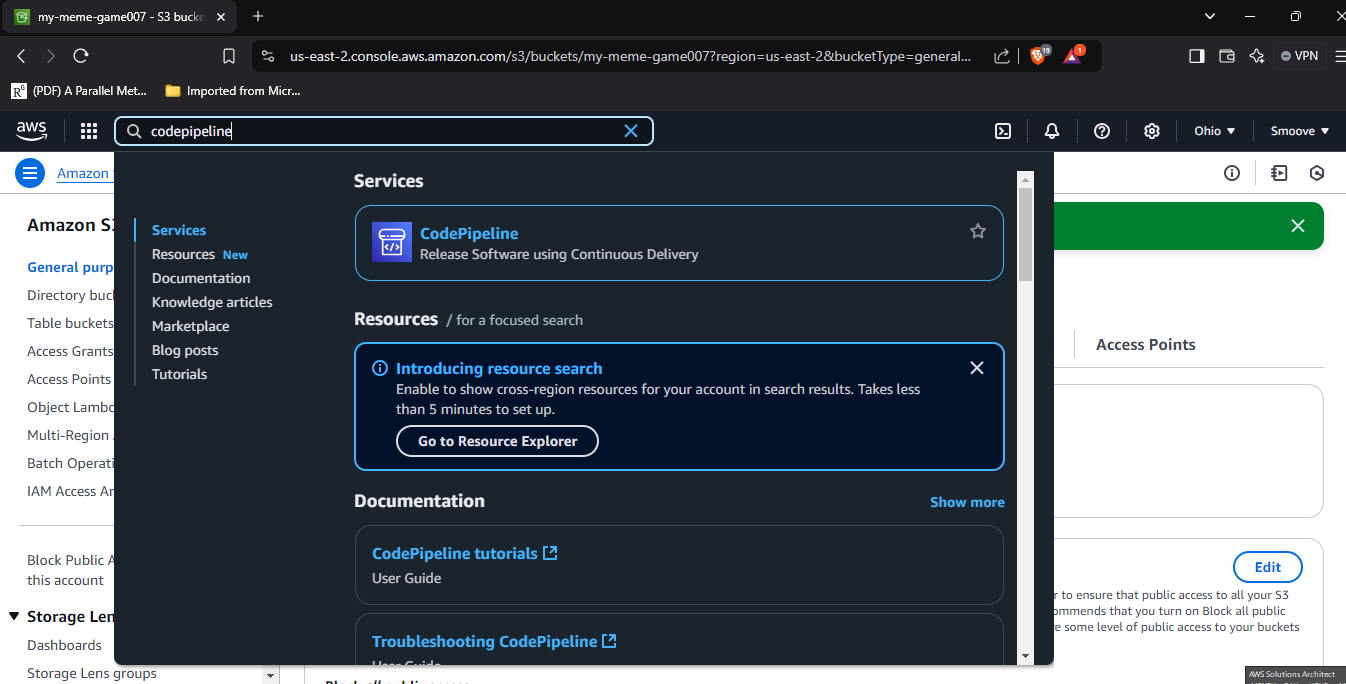


Having done all this successfully, scroll down and click on ‘Save changes’.

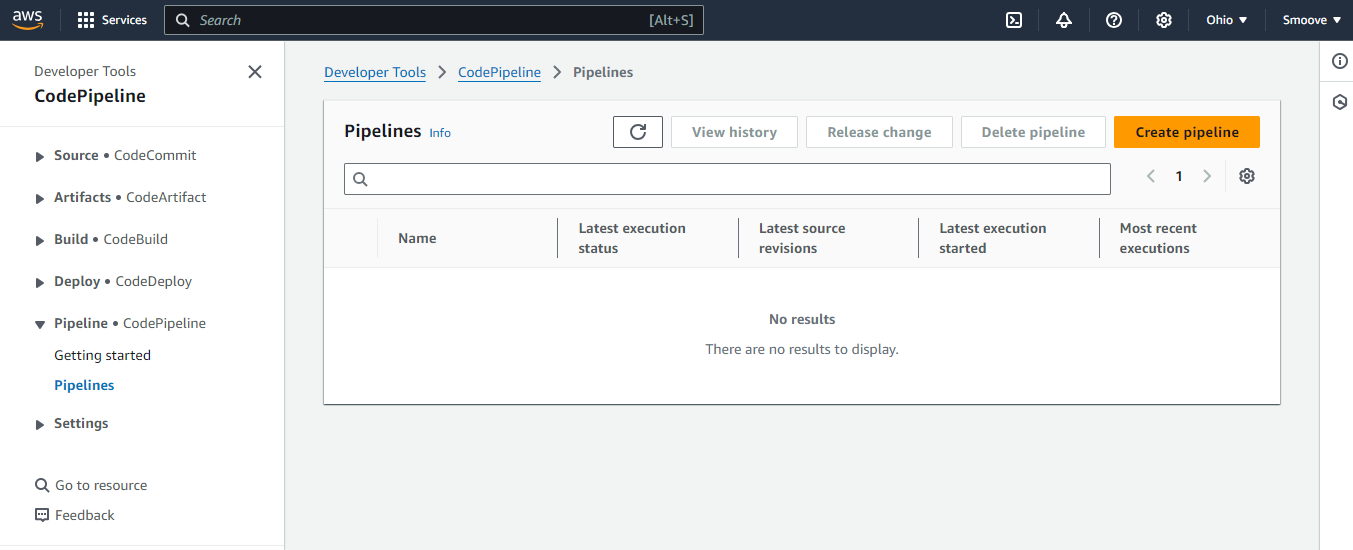


At this point, we have our code stored in GitHub. We’ve also set up our S3 account and its associated permissions and configured it for static website hosting. The next thing we have to do is make use of the CodePipeline service which will take care of getting the code from GitHub into the S3 bucket.

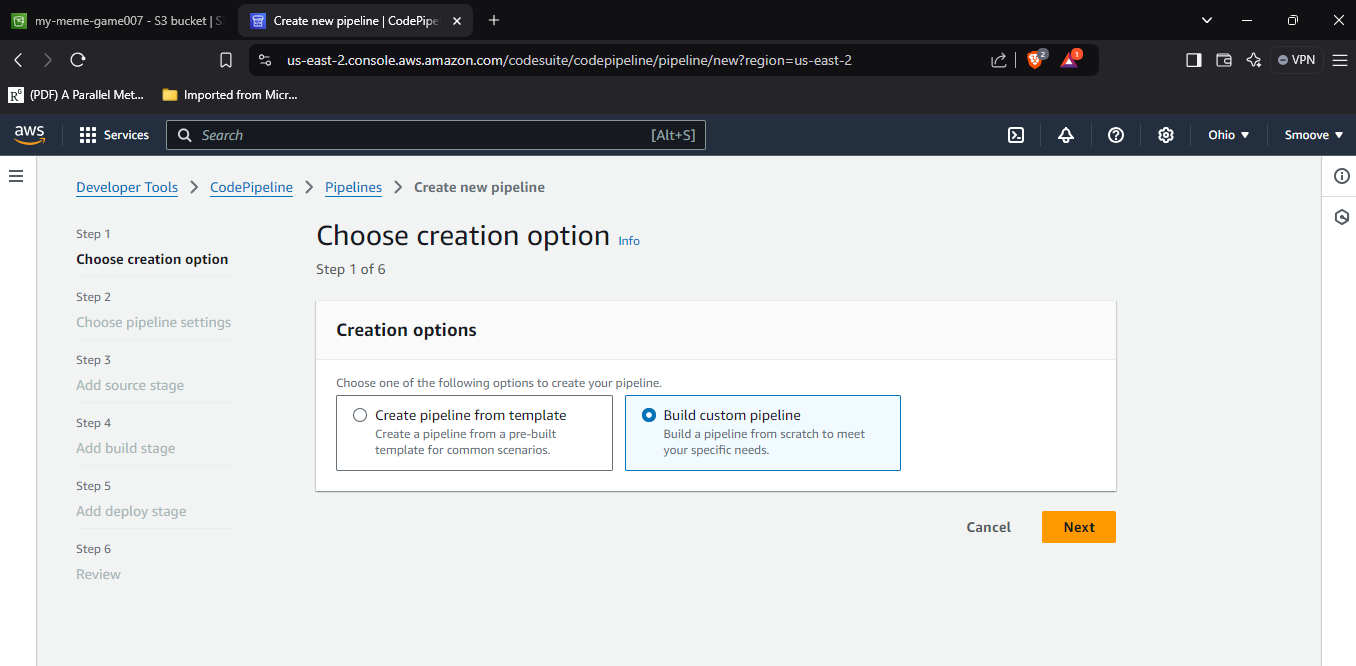
So, let’s go back to our AWS Console and open up a new tab for CodePipeline in the search bar.



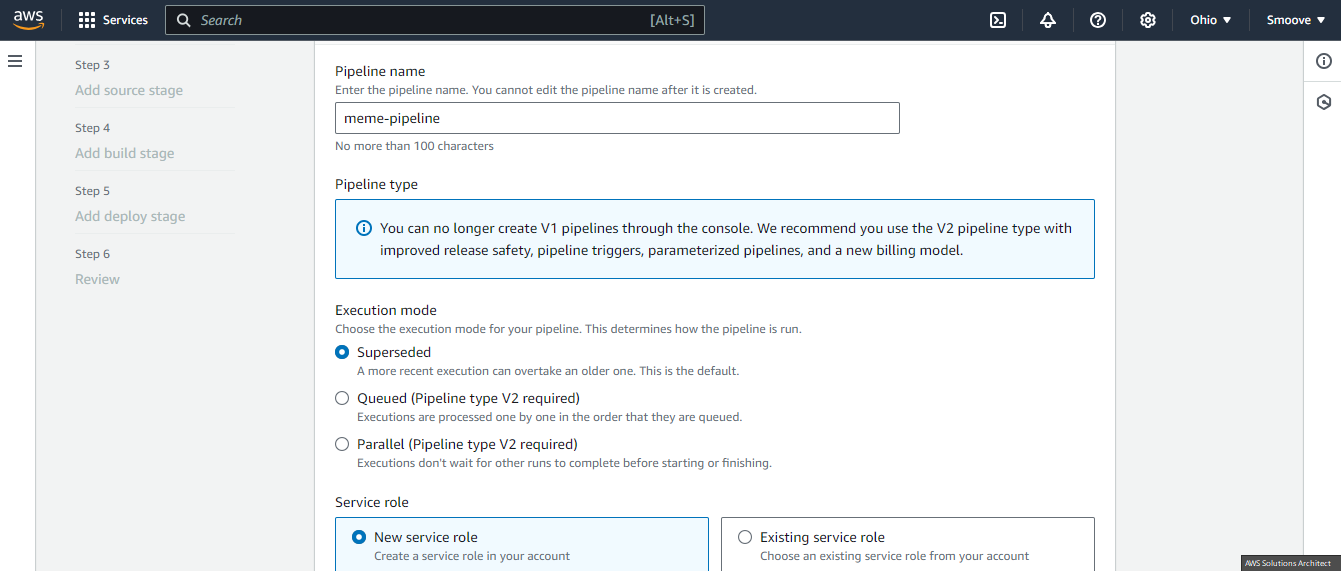
And then we will click on ‘Create pipeline’ on the next page:



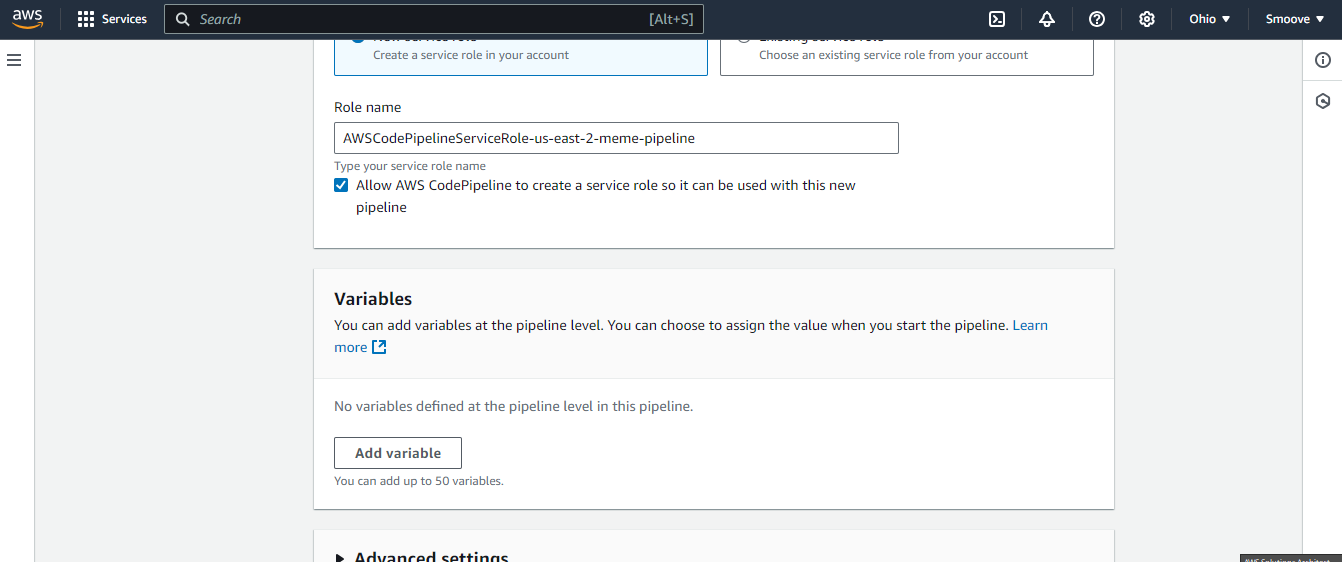
From the options on the next page, choose ‘Build custom pipeline’ and click on ‘Next’.



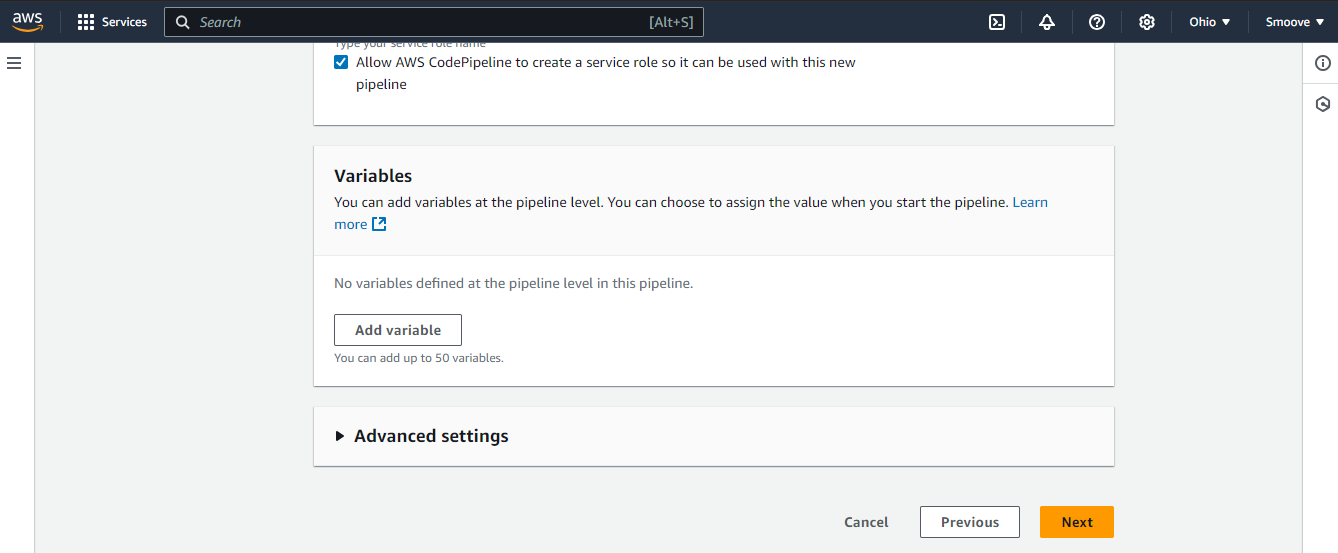
From the screenshot below, we will observe that version 1 (V1) is no longer available, so we go ahead and use V2. We give it a consistent name, in this case, ‘meme-pipeline’, and then we select the Execution mode and service roles as shown.

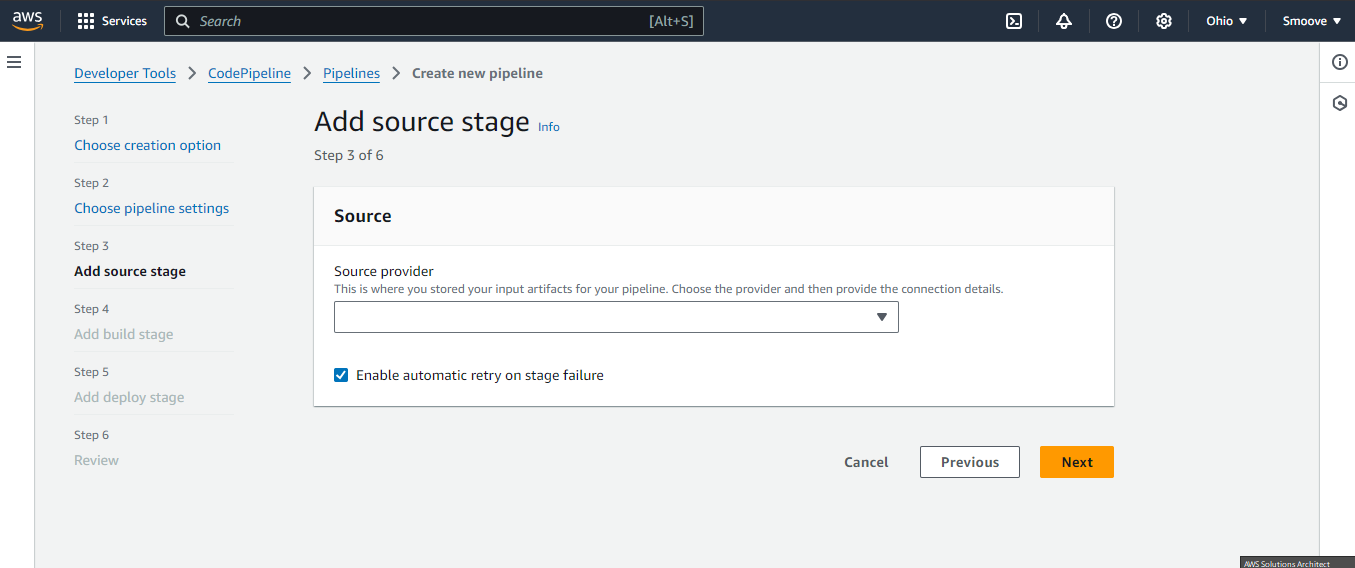


Scroll down, and in the ‘Role name’ section, you will observe that a name has been automatically filled for you.

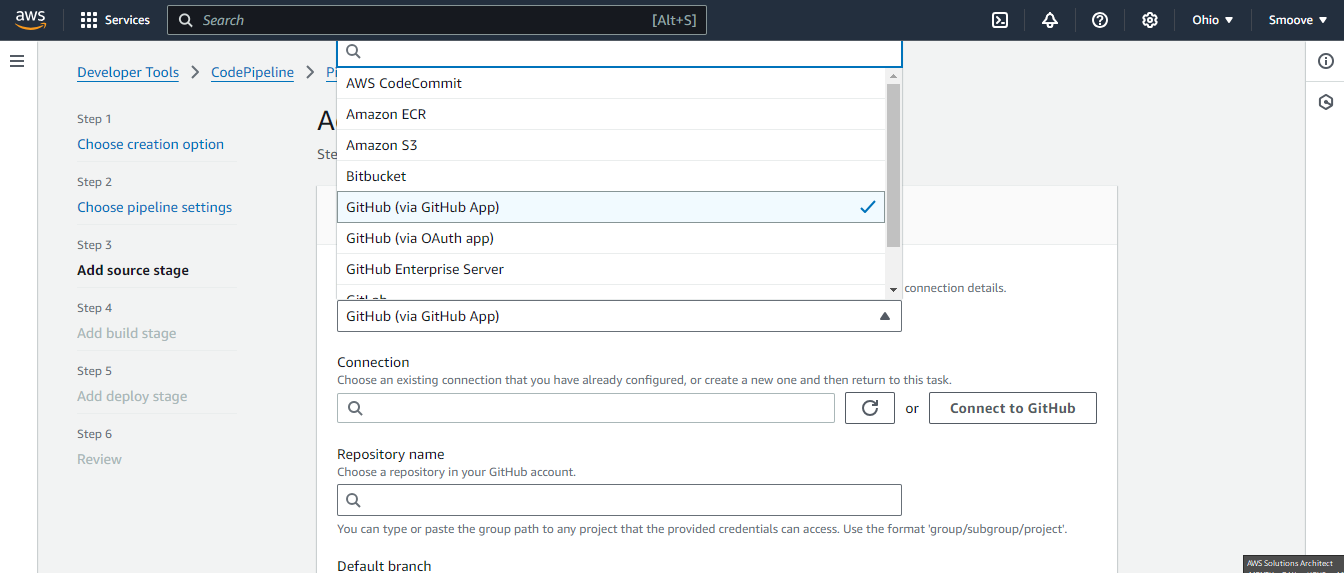


Go ahead and leave everything else as-is, and click on ‘Next’.

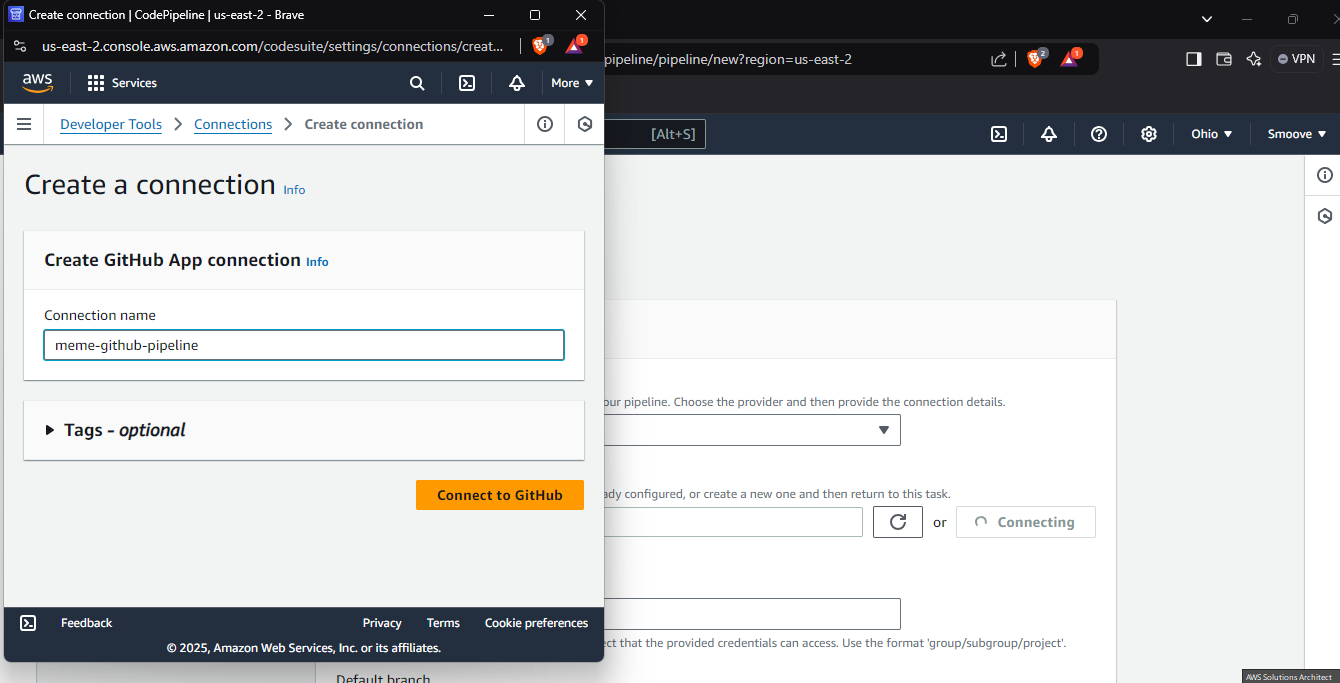


The next page should look something like this:  


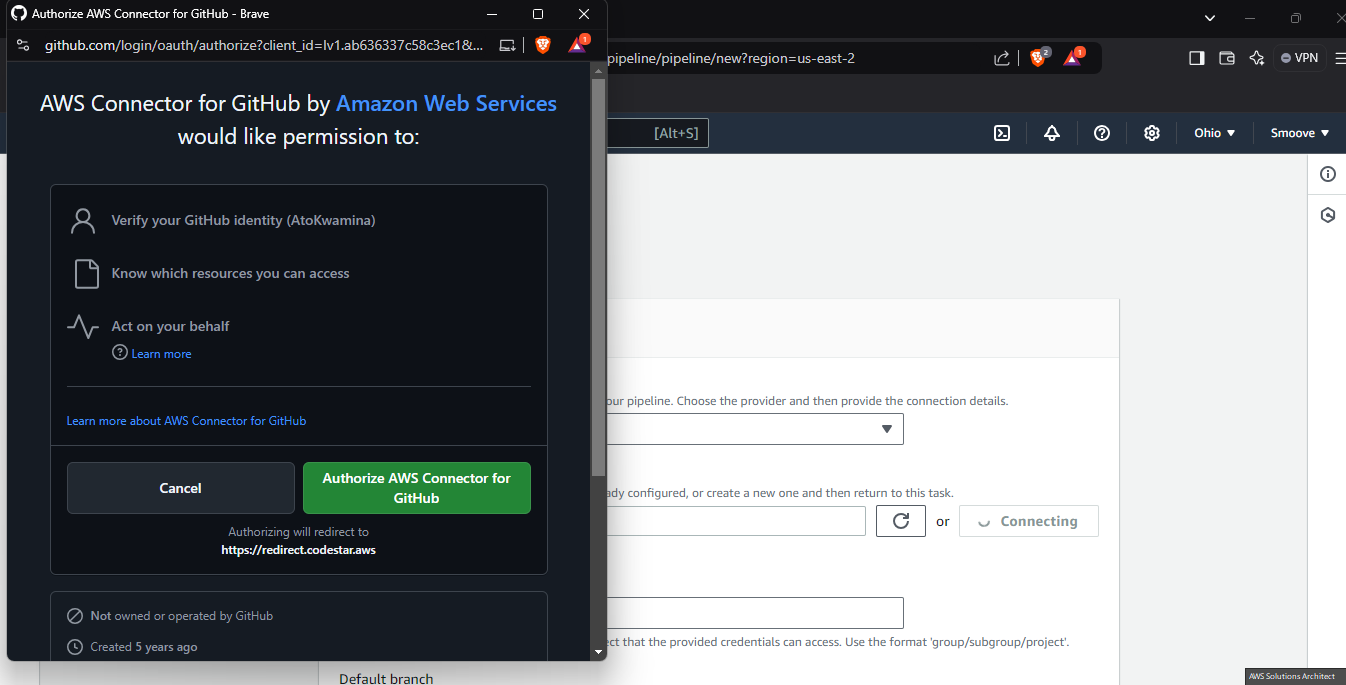
Click on the drop-down menu under ‘Source provider’ and select ‘GitHub (via GitHub App)’. Then, click on ‘Connect to GitHub’:



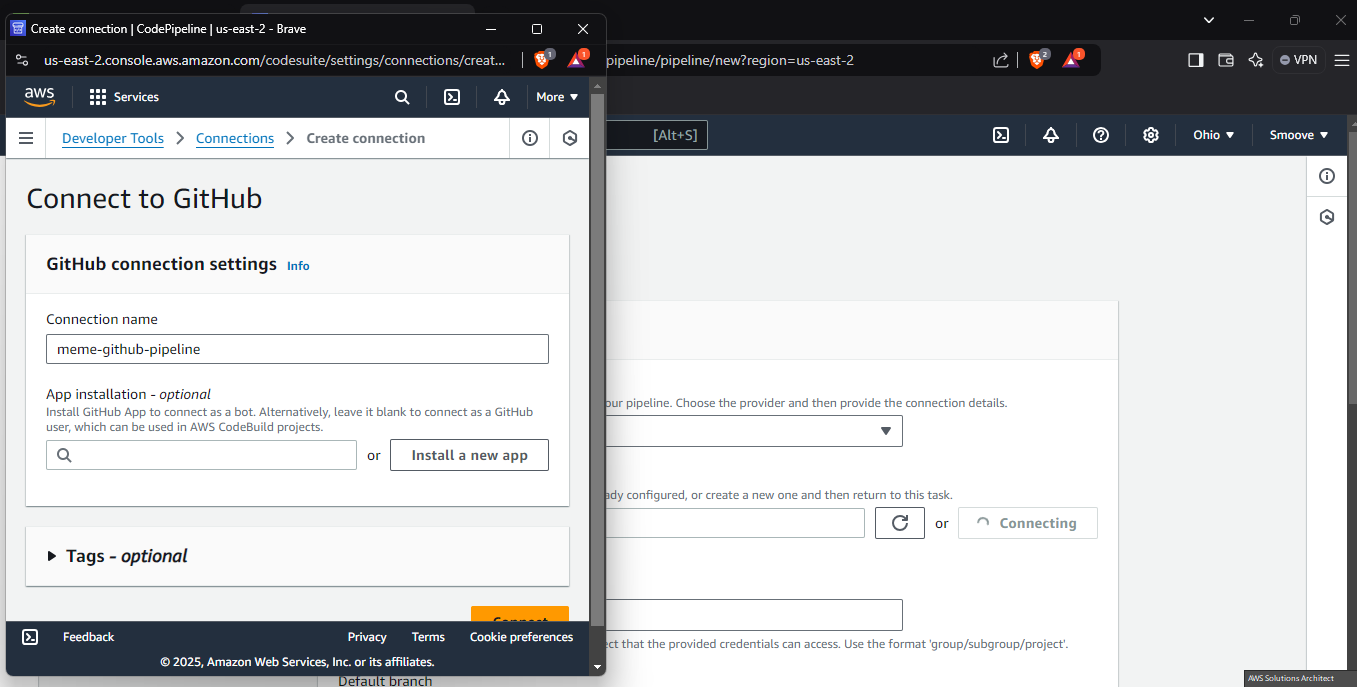
A pop-up tab opens. Type a connection name; in my case I am using ‘meme-github-pipeline’. Then click on ‘Connect to GitHub’.



You should then see a window like this:

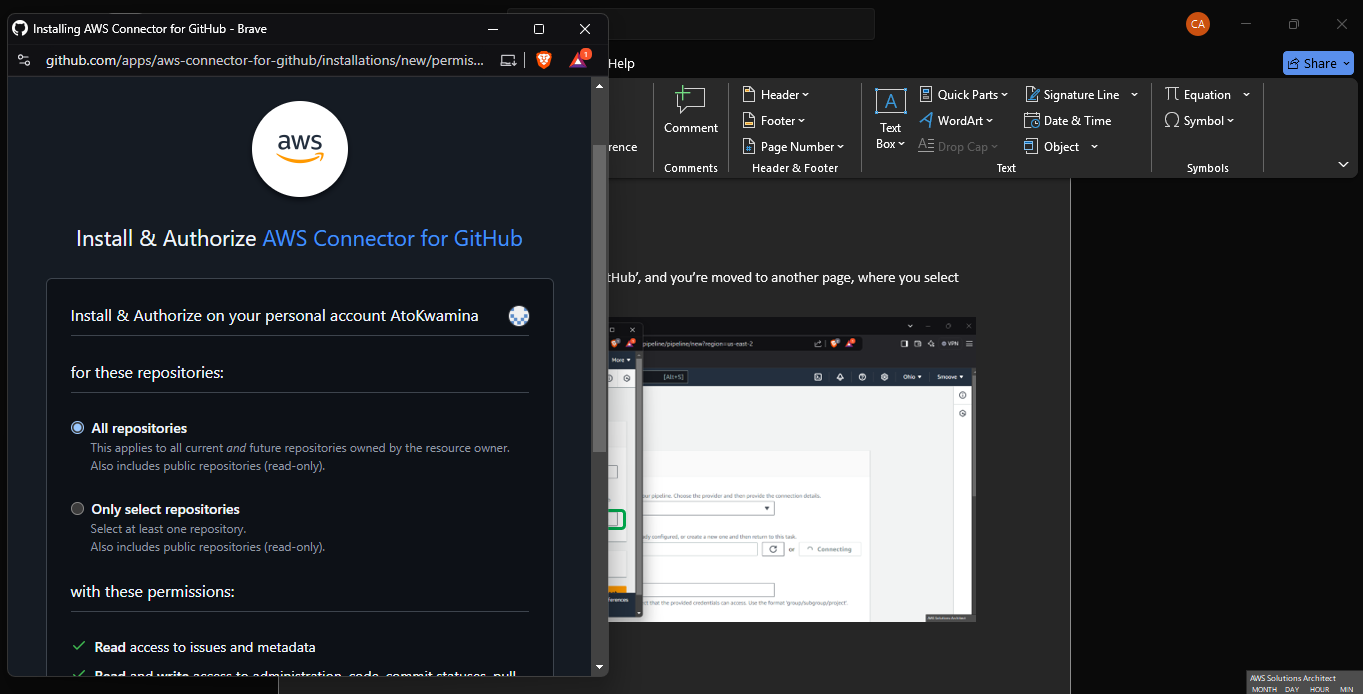


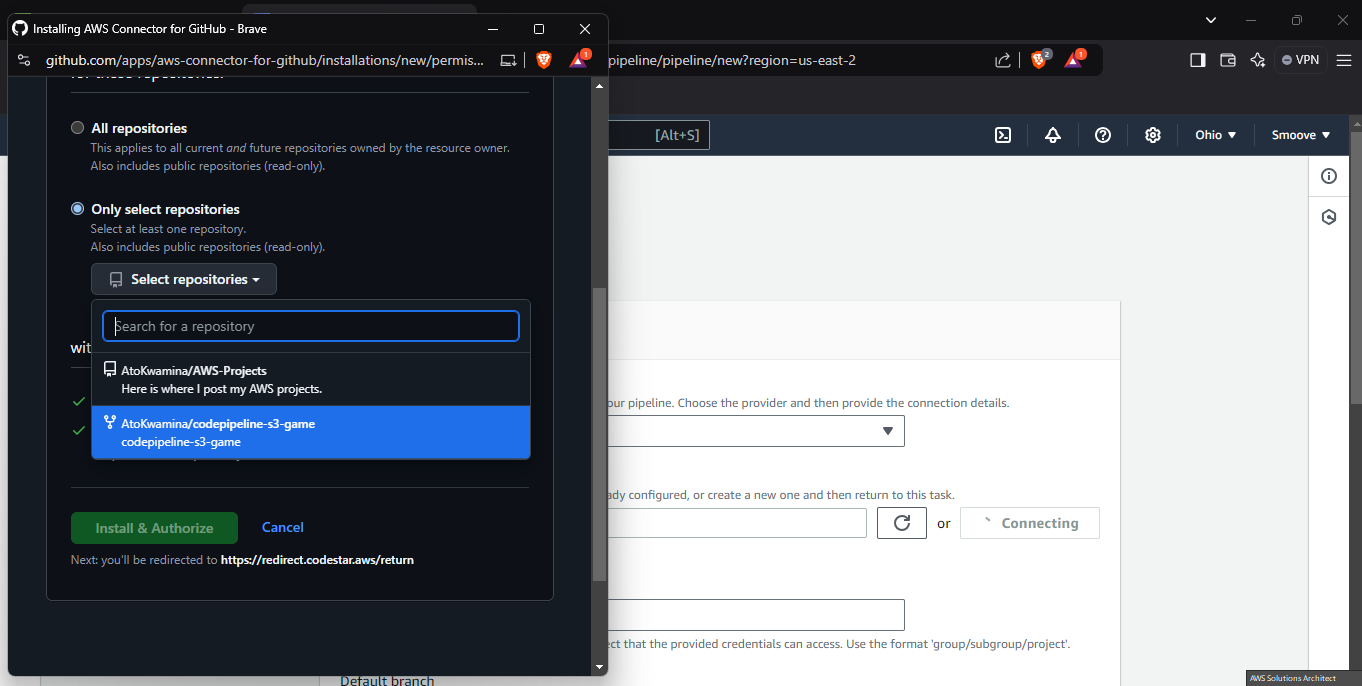
Select ‘Authorize AWS Connector for GitHub’, and you’re moved to another page, where you select ‘Install a new app’:



A new page opens like this. As seen from the screenshot below, my own GitHub account is showing now. If done correctly, your own GitHub account should also be displayed.

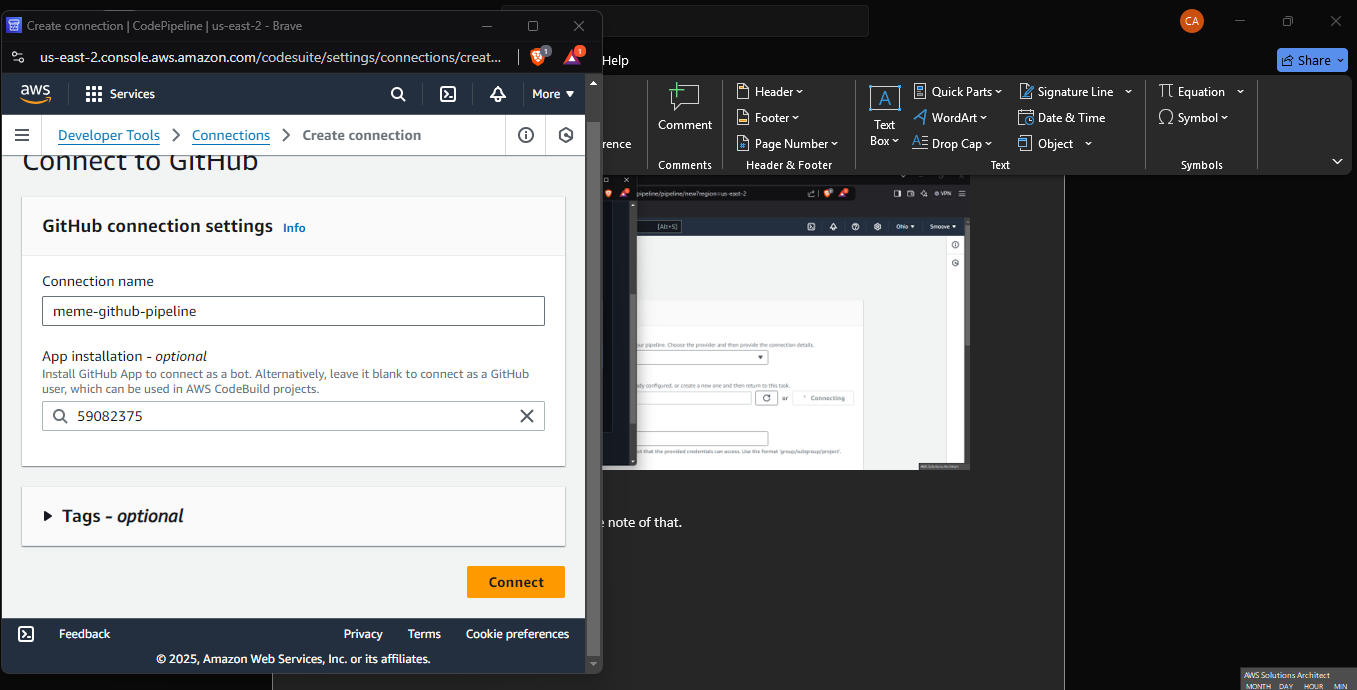
Choose ‘Only selected repositories’, and then from there, select the repository that was forked into your GitHub account. Finally, click on ‘Install and Authorize’.



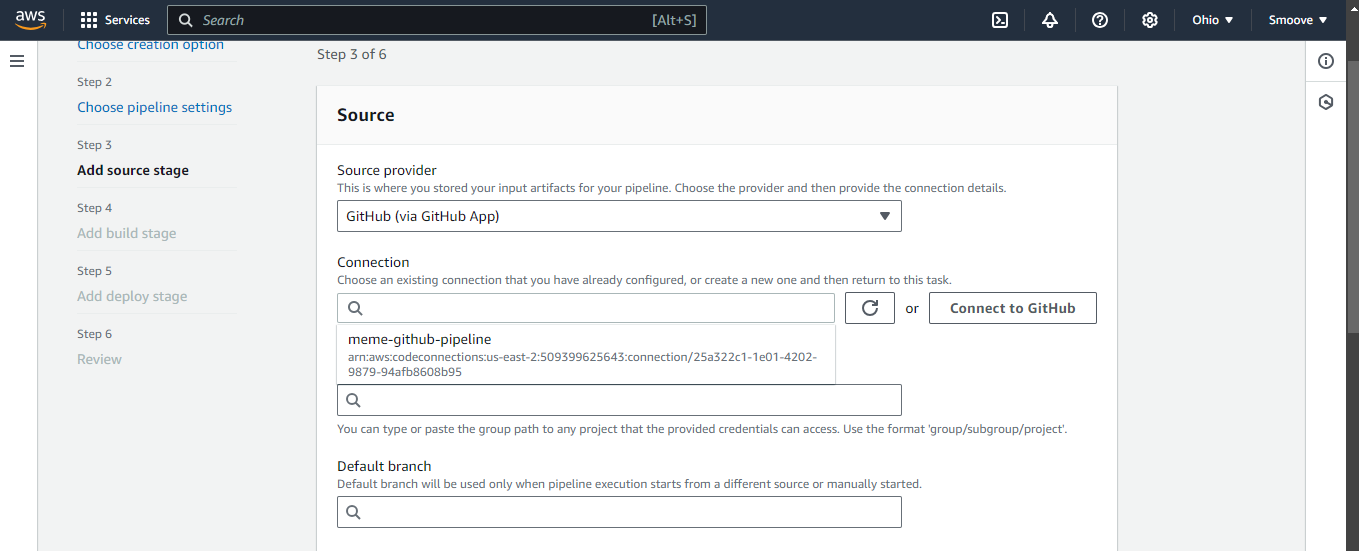


NB: You may be asked to sign in, so take note of that.

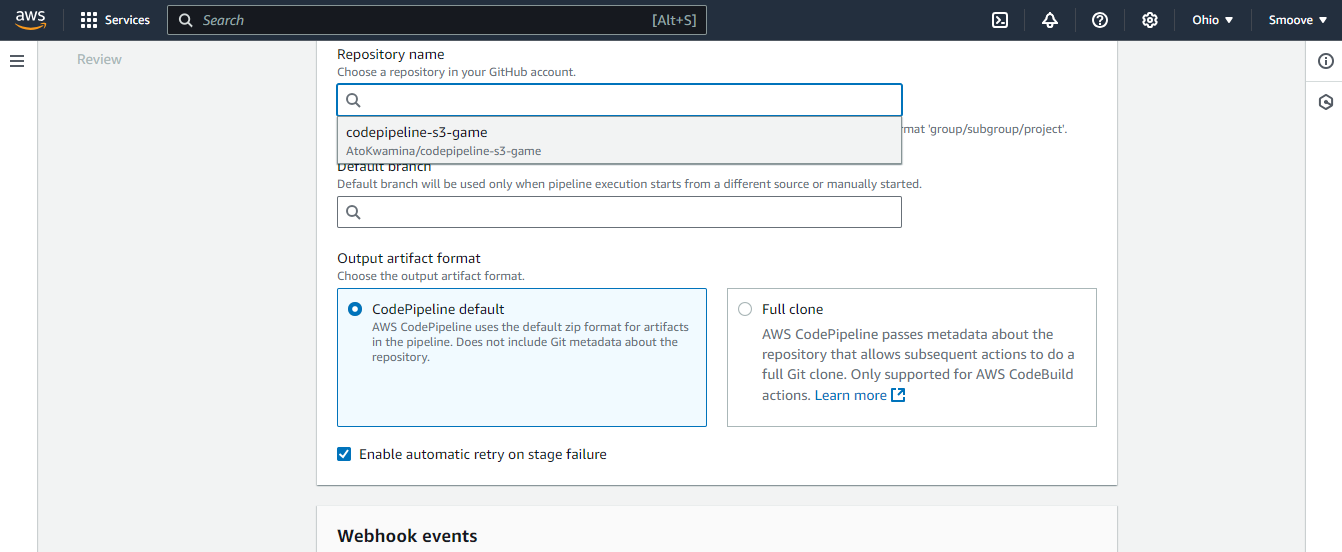
On the next page, go ahead and click on ‘Connect’.



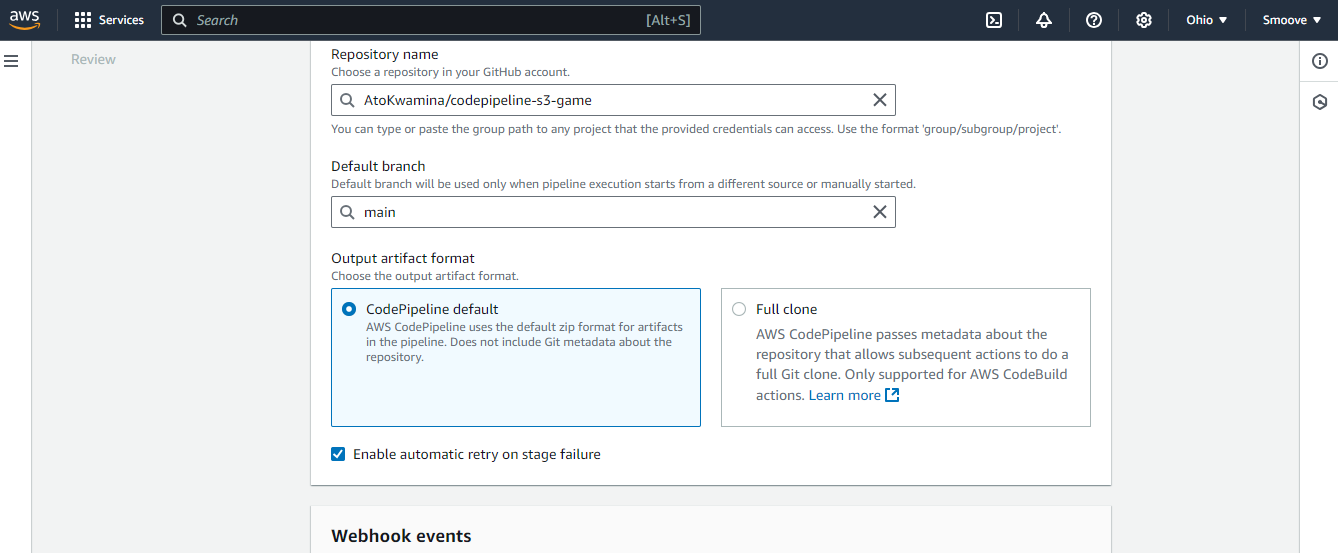
Once successful, the connection you just created should be present in the ‘Connection’ tab. Click on the tab and select it as I have selected mine below.

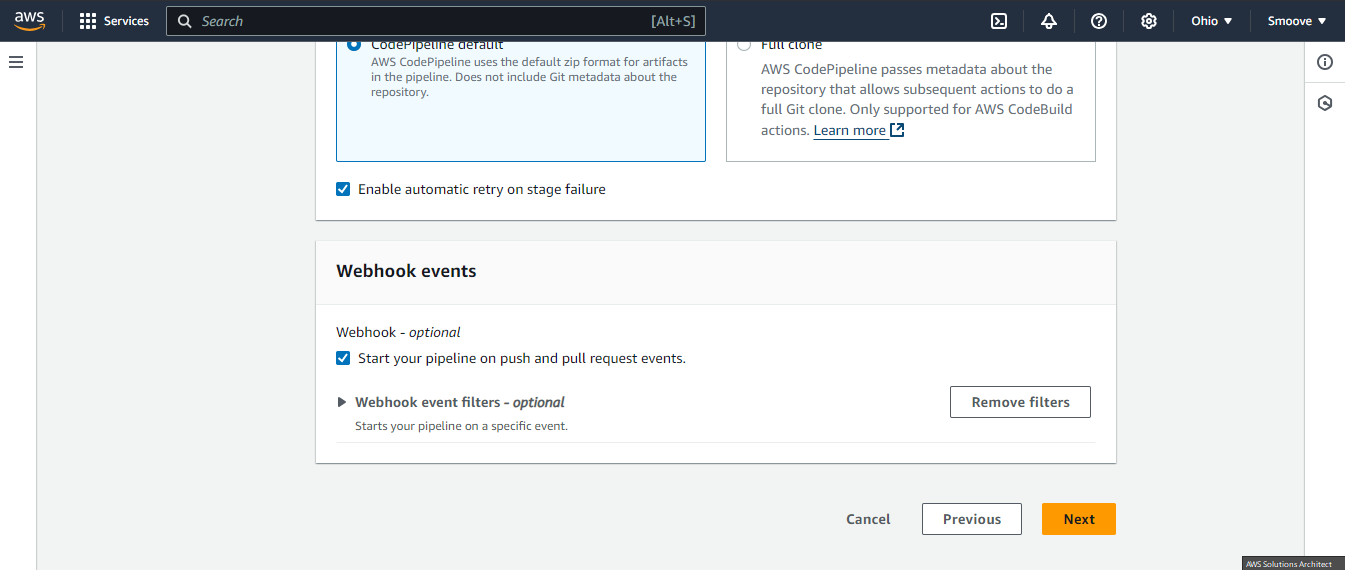


Under the ‘Repository name’ option, select the name that pops up (pertaining to what you just created).

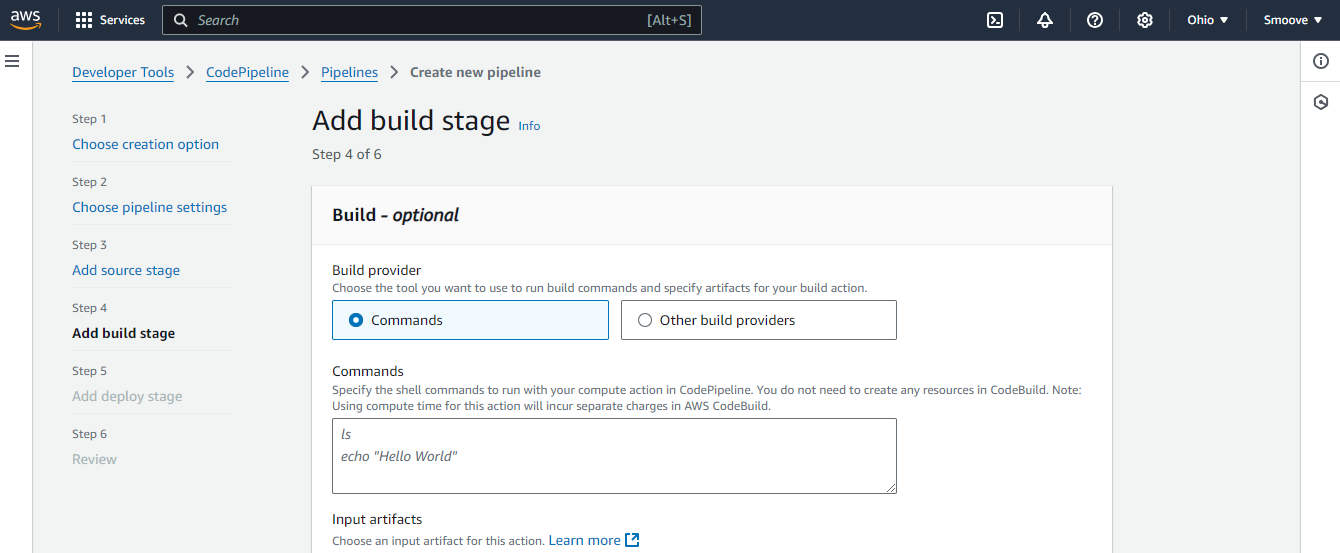


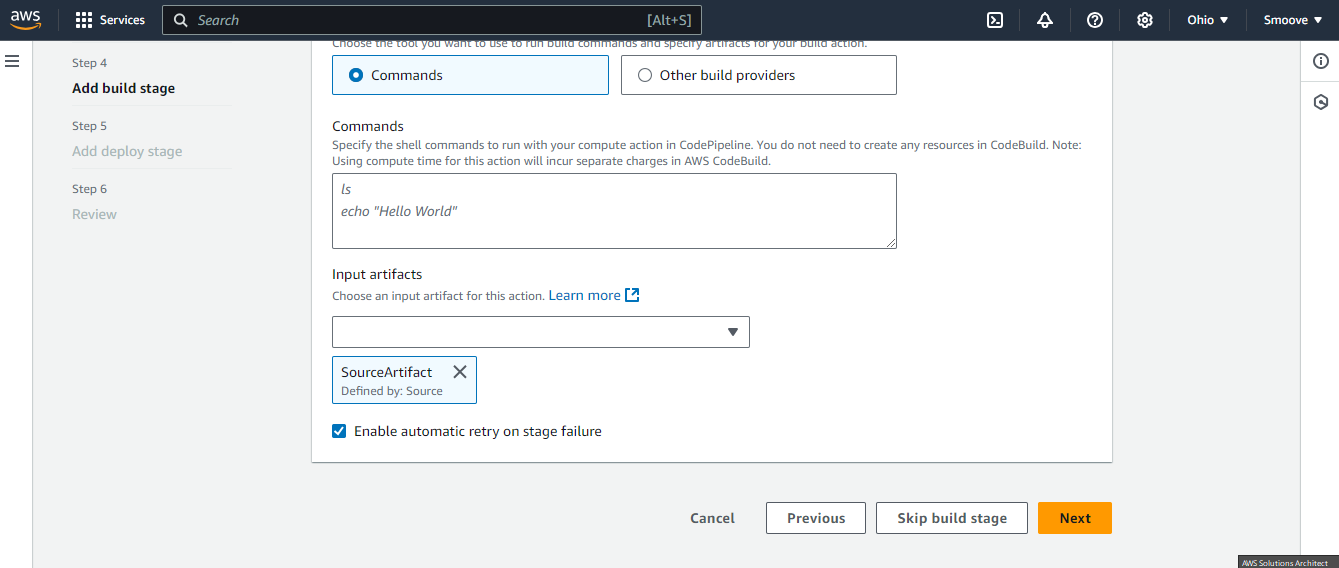
Use the default branch name ‘main’, then select ‘CodePipeline default’ and click on ‘Next’.



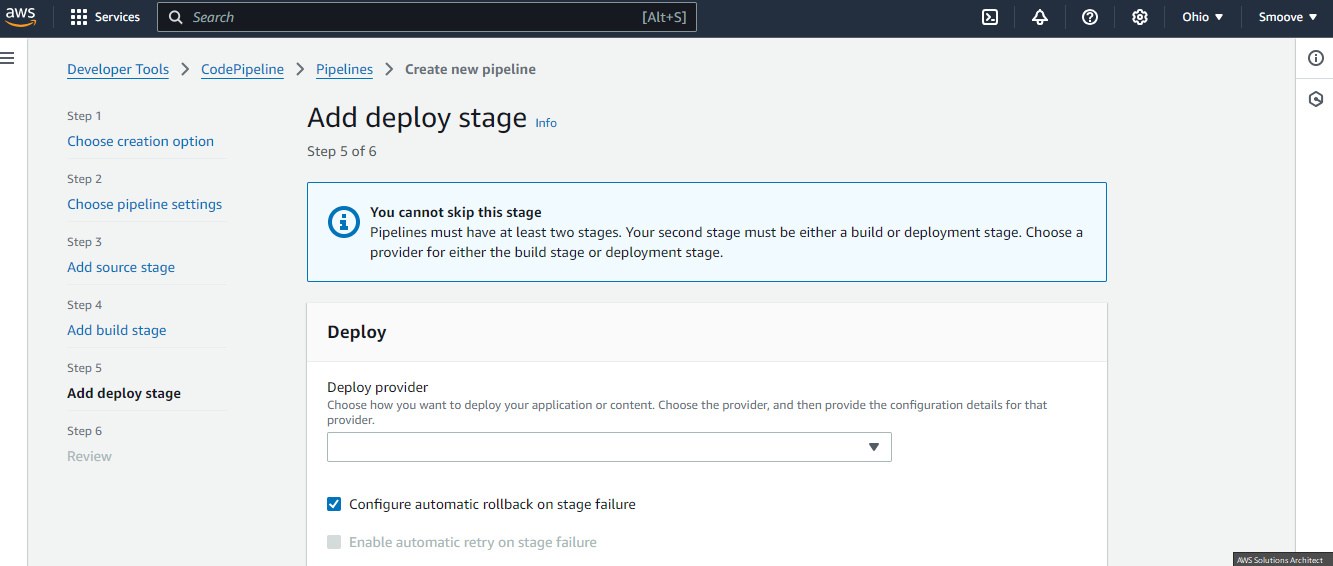


For the ‘Build stage’ option, we will skip it since it relates to the part of the exercise on build and test, which we will not be doing here.

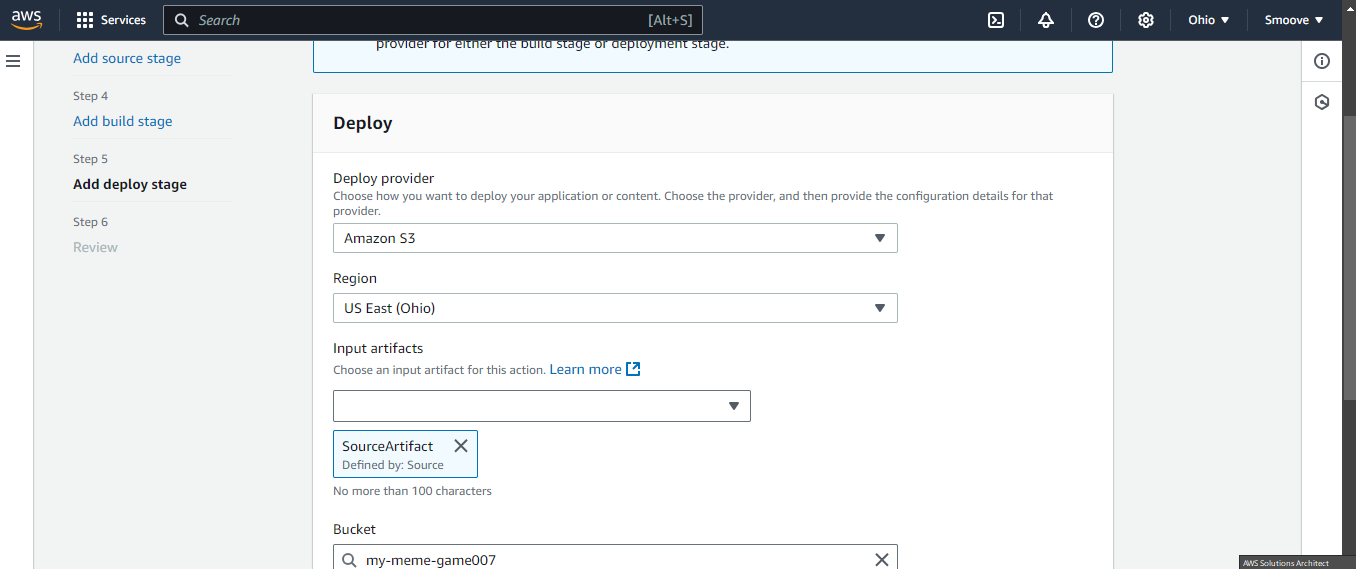




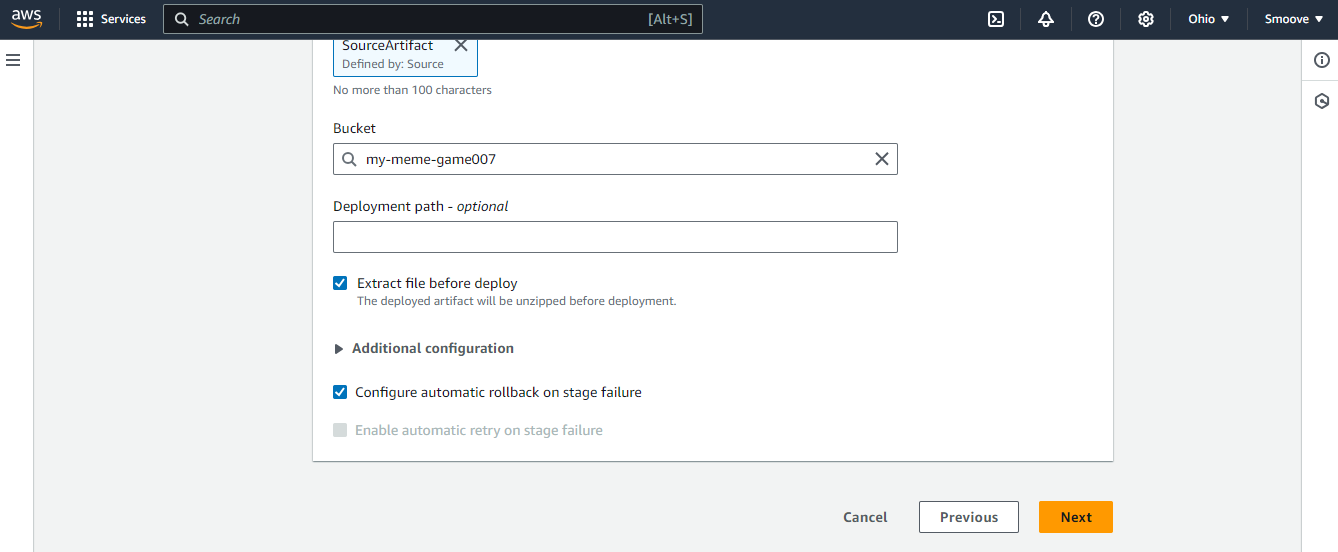
On the next page, we have the ‘Add deploy stage’ option. This is where we choose the service in which we will be deploying our code.



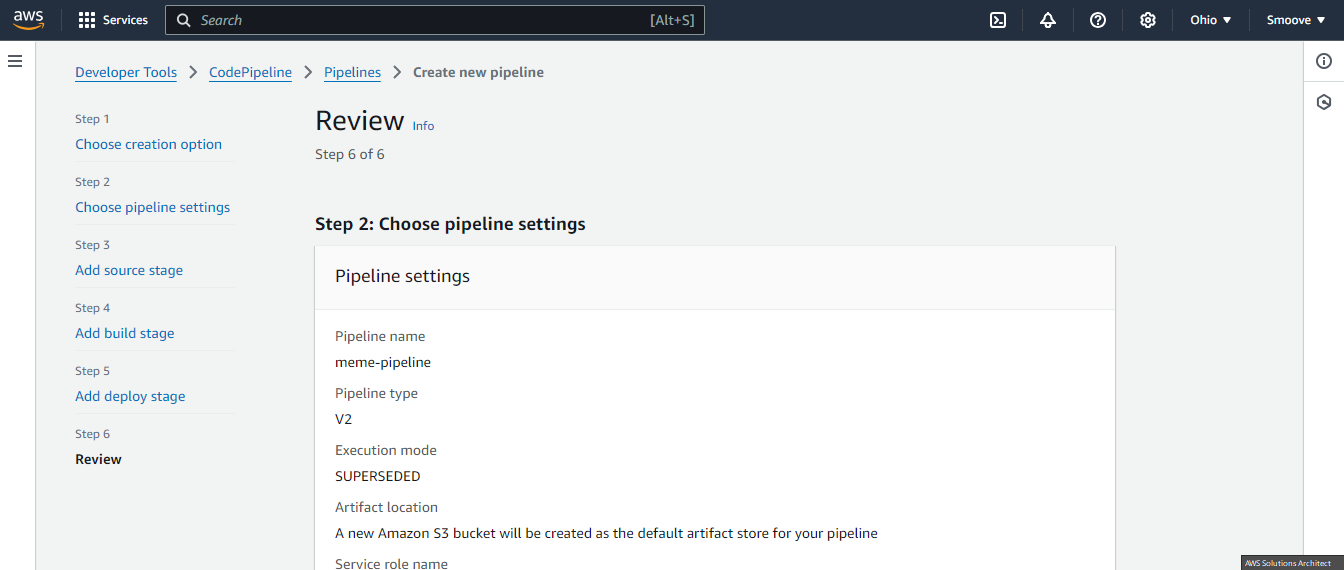
Under the ‘Deploy provider’ side, select ‘Amazon S3’. Then ensure the region is the region you started with and are currently in, and also select the bucket (your bucket will automatically be an option once you click in the option bar).

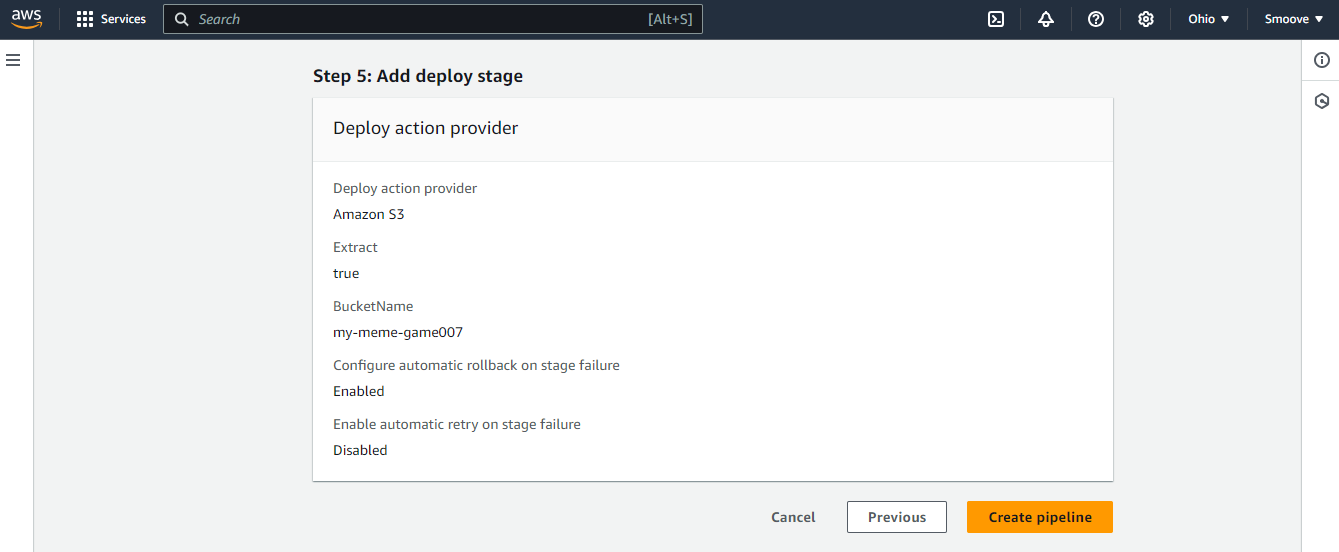


Make sure the ‘Extract before deploy’ option is selected to get rid of the ‘S3 object key’ option, as we won’t be needing that. Then click ‘Next’.

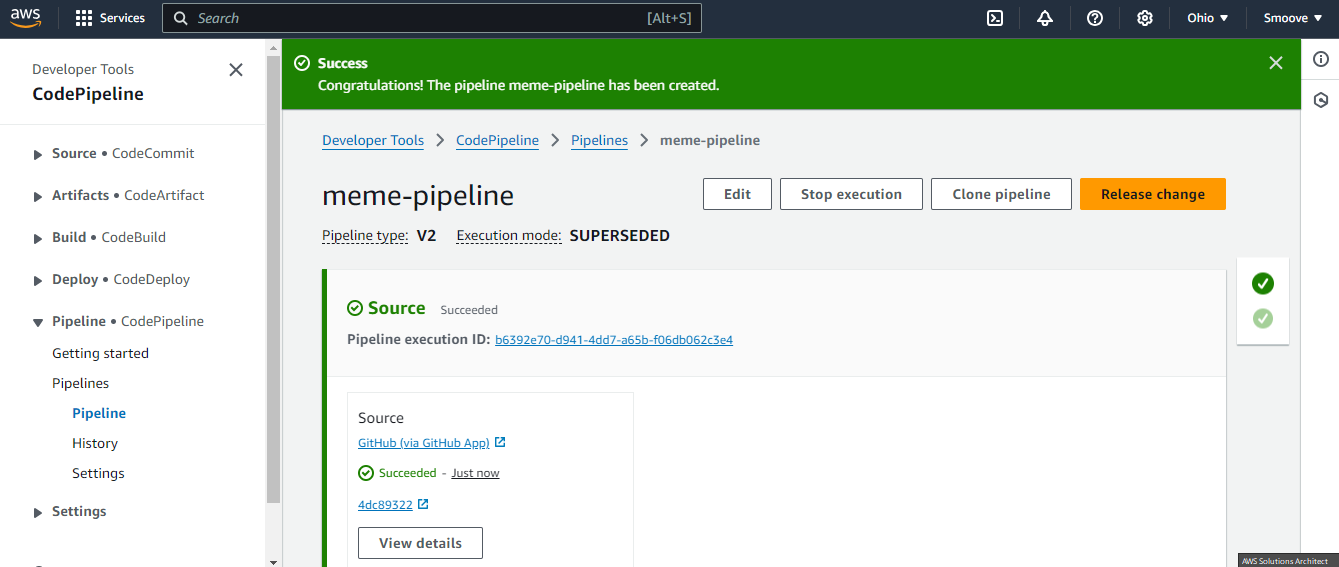


The next page you will see is the ‘Review’ page. Ensure you have set everything up properly here, then click ‘Next’.

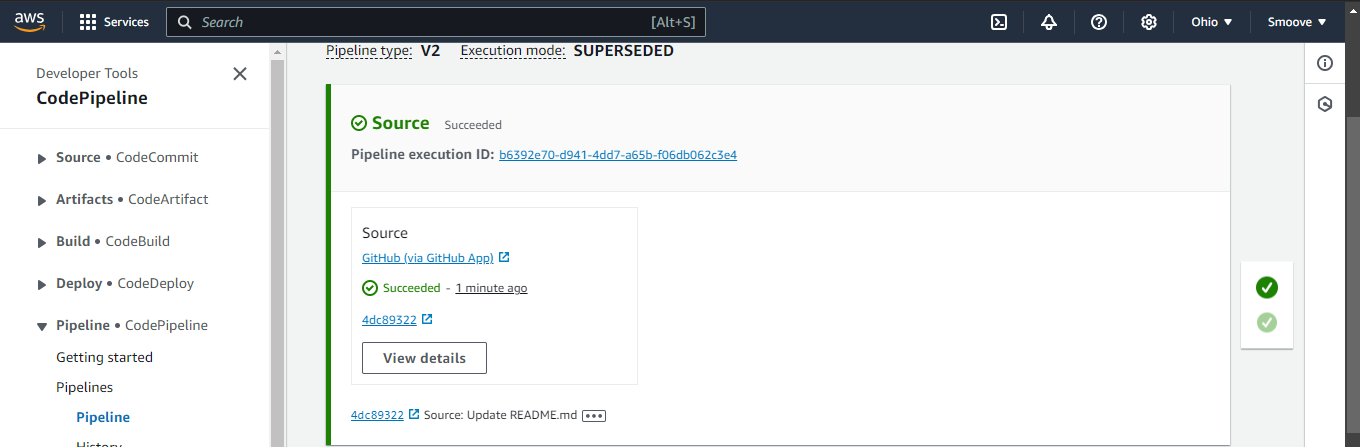




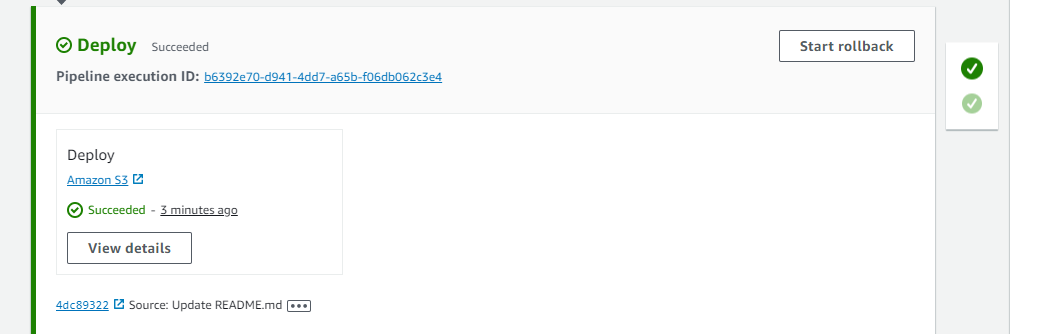
And there you have it! Pipeline created.



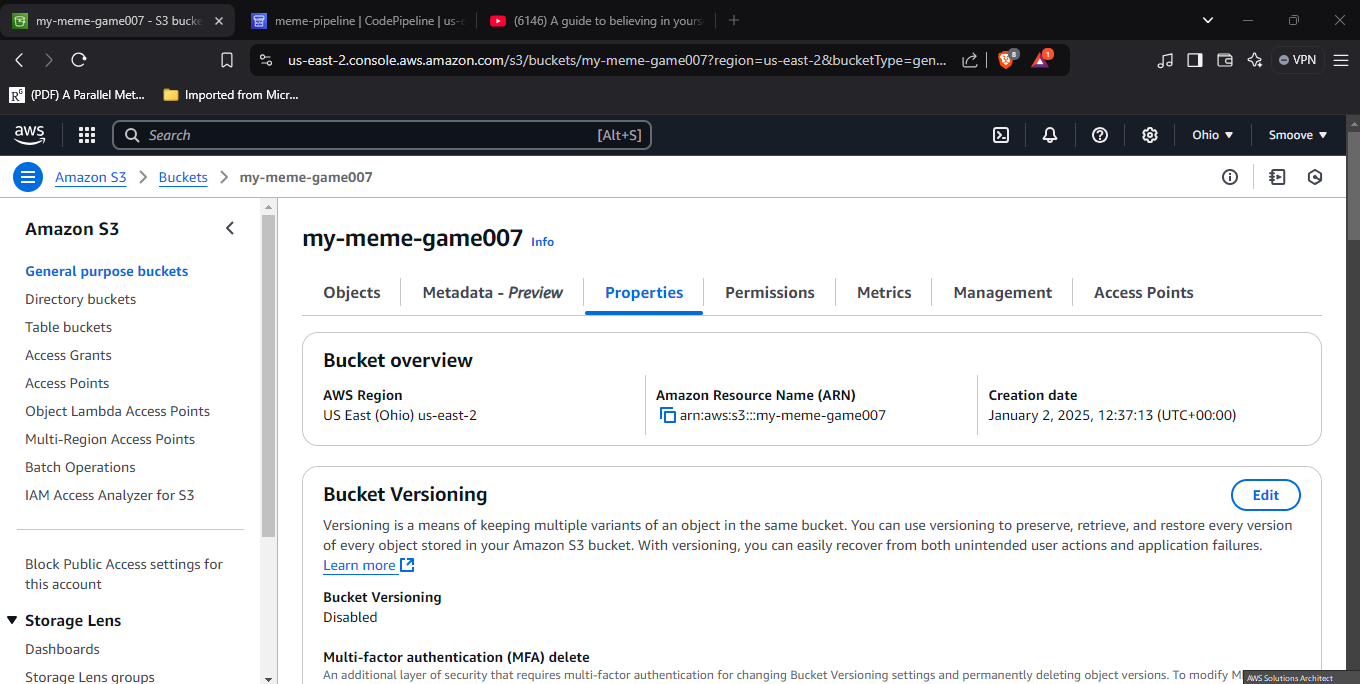
So, you will observe two parts to this. The first part is where CodePipeline grabs the code needed from our GitHub repository:



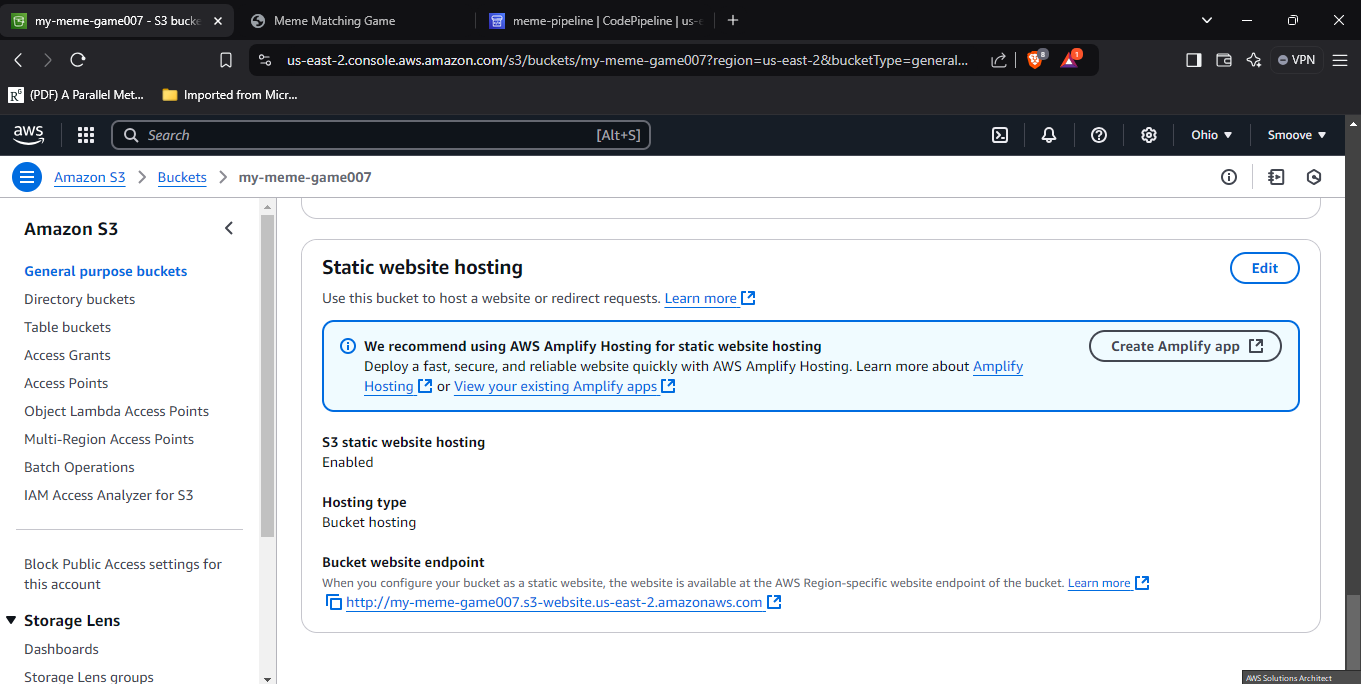
The second part will be where CodePipeline then deploys the code on S3:



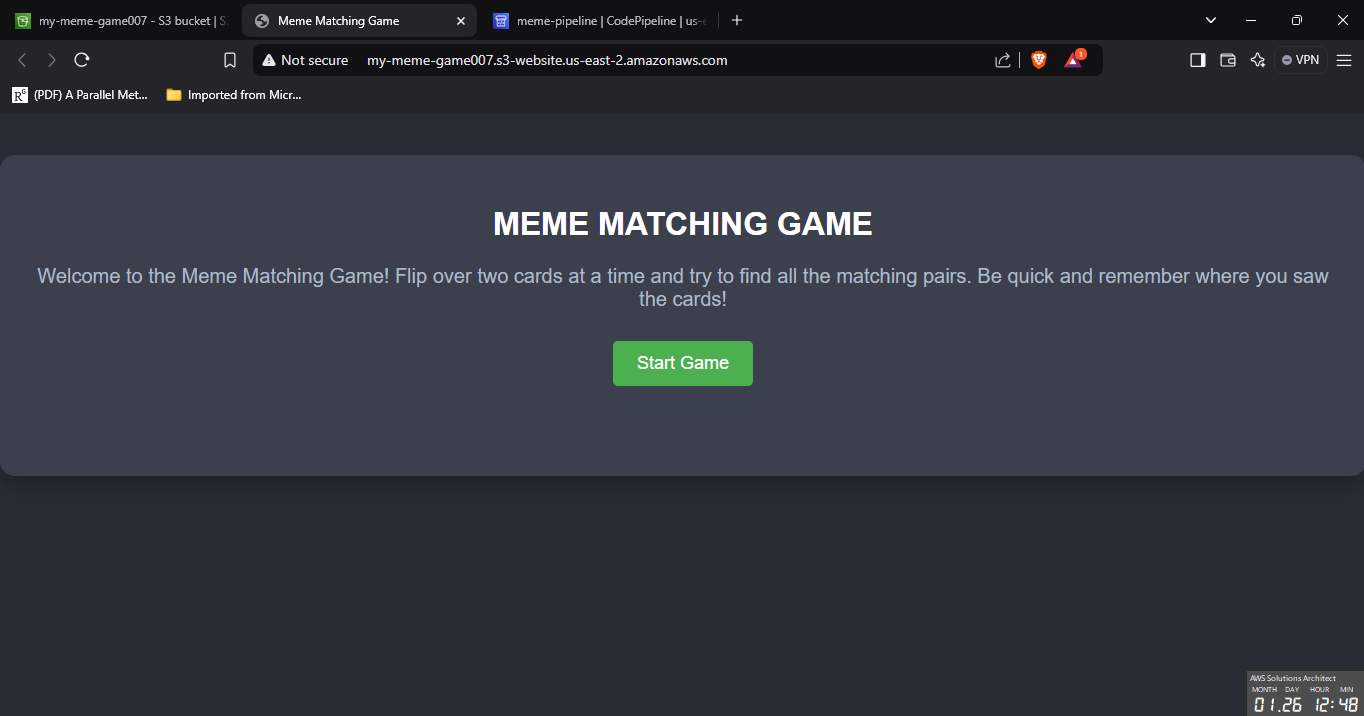
We can now go ahead and test things out. We will start by going back to the S3 tab and choosing ‘Properties’:

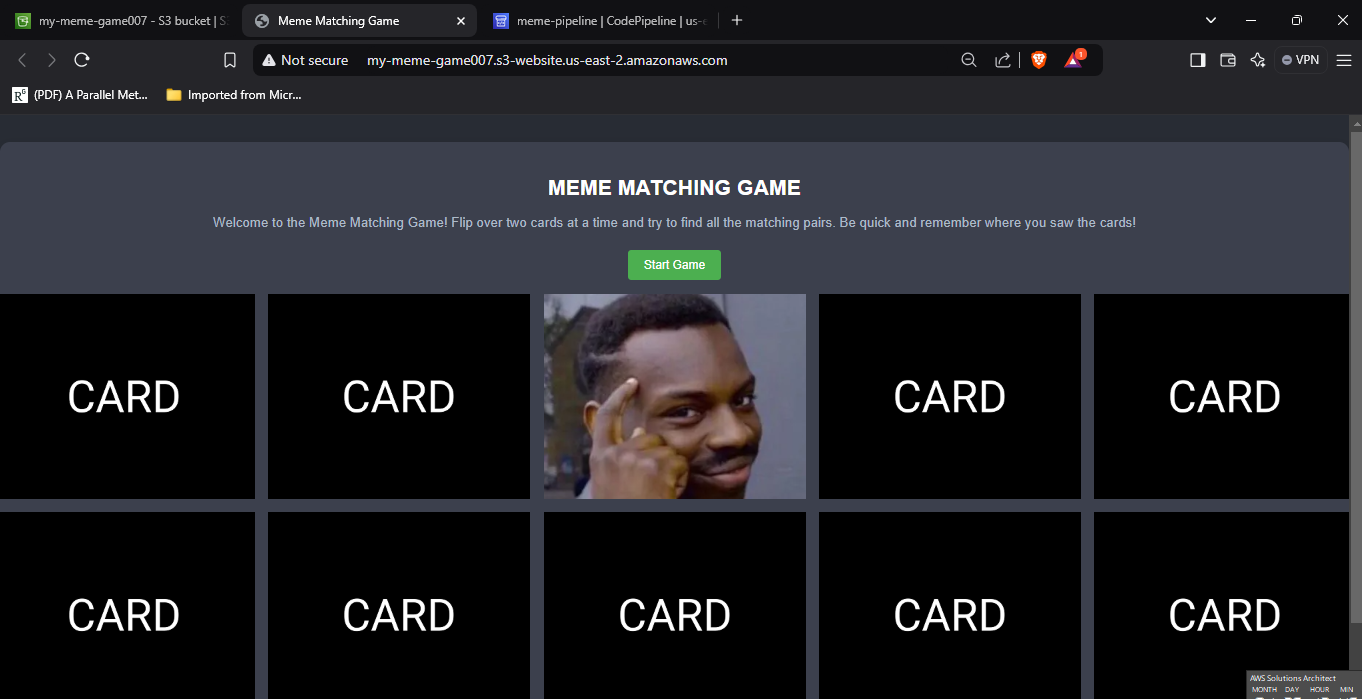


Scroll all the way down to ‘Static website hosting’ section. There should be a bucket website endpoint there. Click on it to reveal the result of your code.

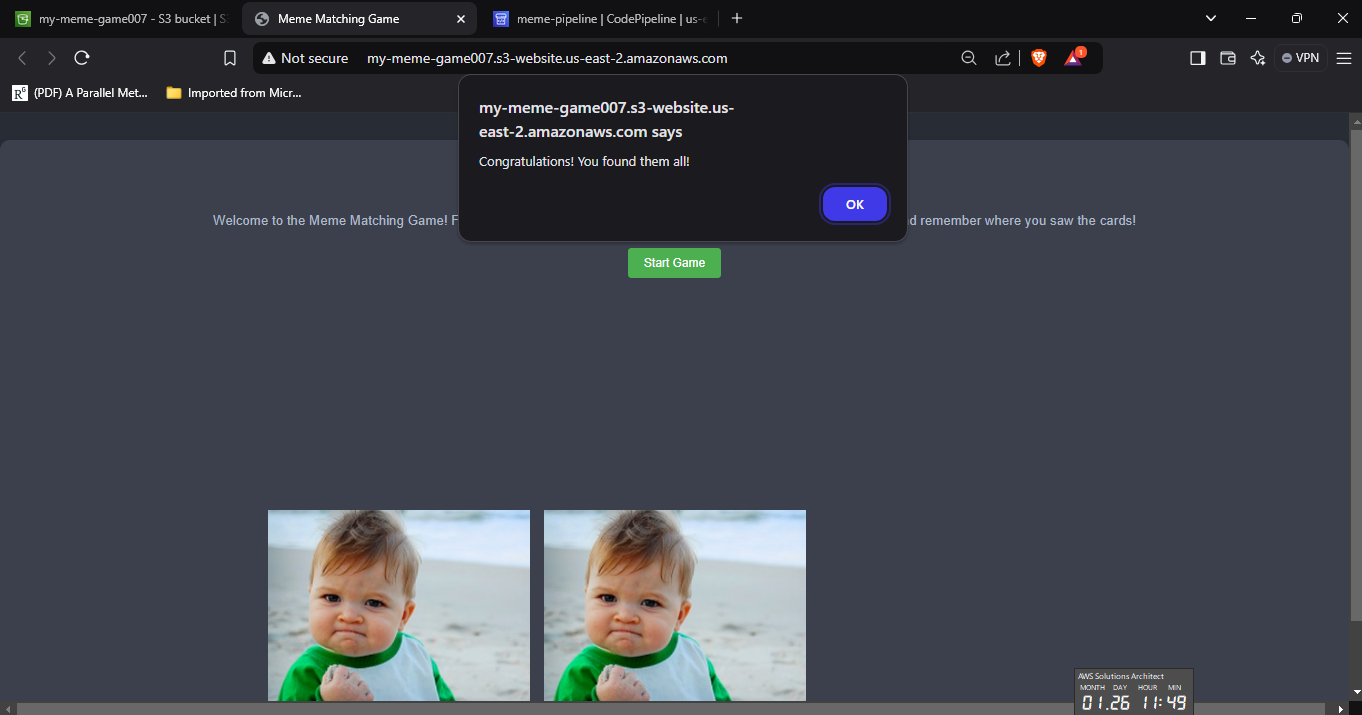


And there you have it!



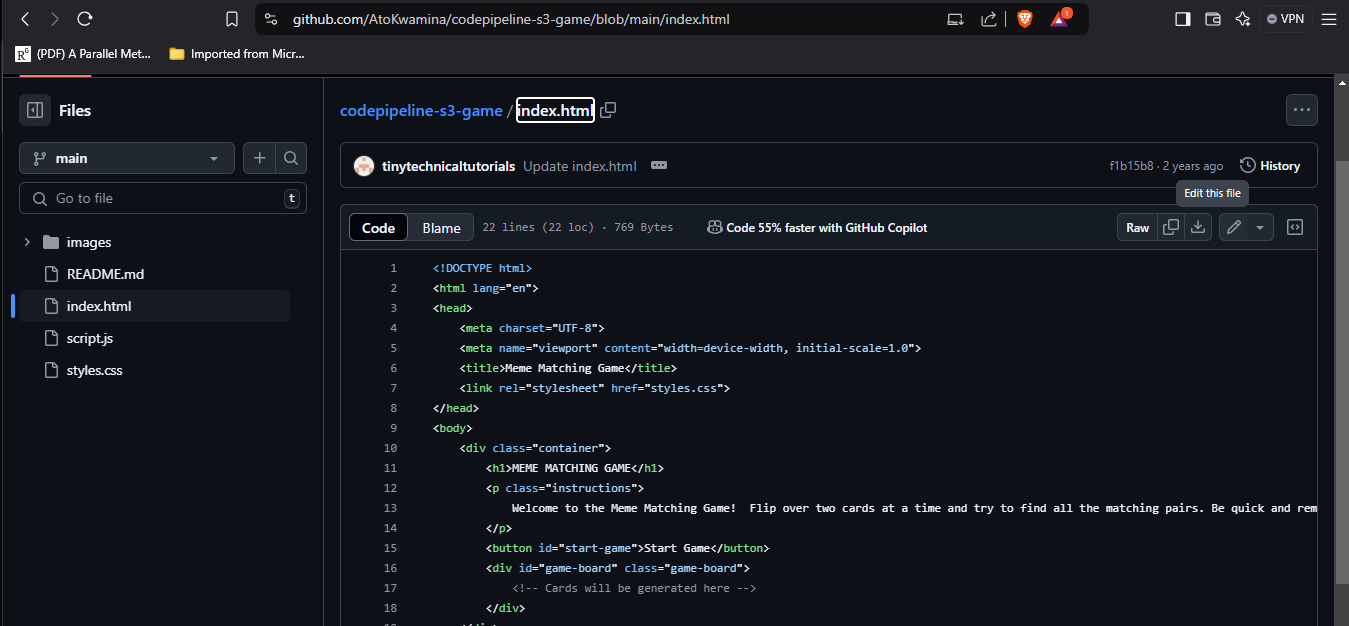


Looks like I am done with my game now. LOL

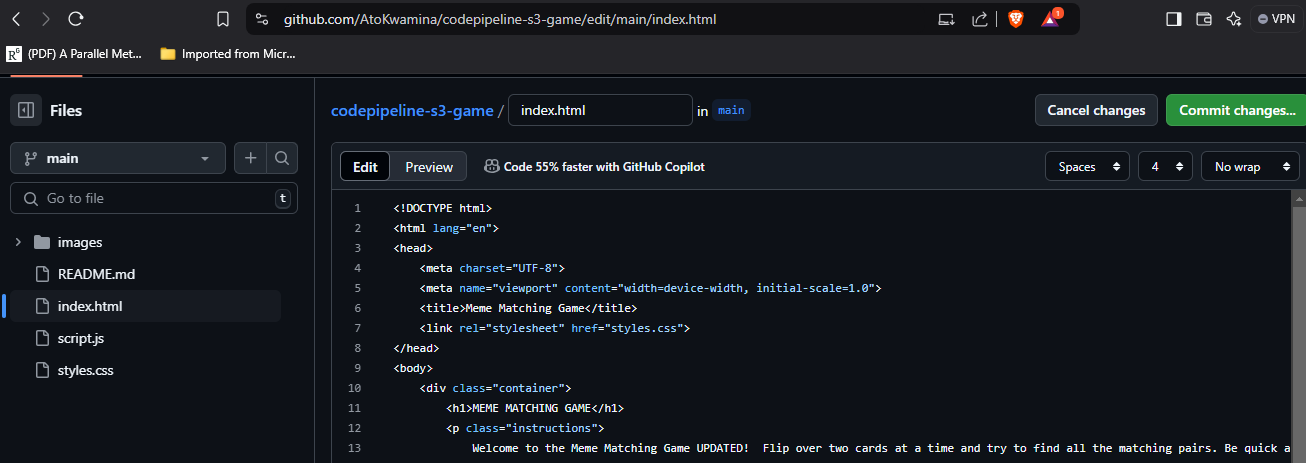


Oh and, one thing we need to remind ourselves of, where CodePipeline is concerned. Remember it’s not just taking code from GitHub and deploying it in S3 per our configuration. It’s also supposed to detect changes in our code and adjust the final product as such. So, let’s test this.

Let’s go back to GitHub and make a small change to our HTML code. Select the index.html file from the options to the left, and click on the pencil icon on the right to edit it.



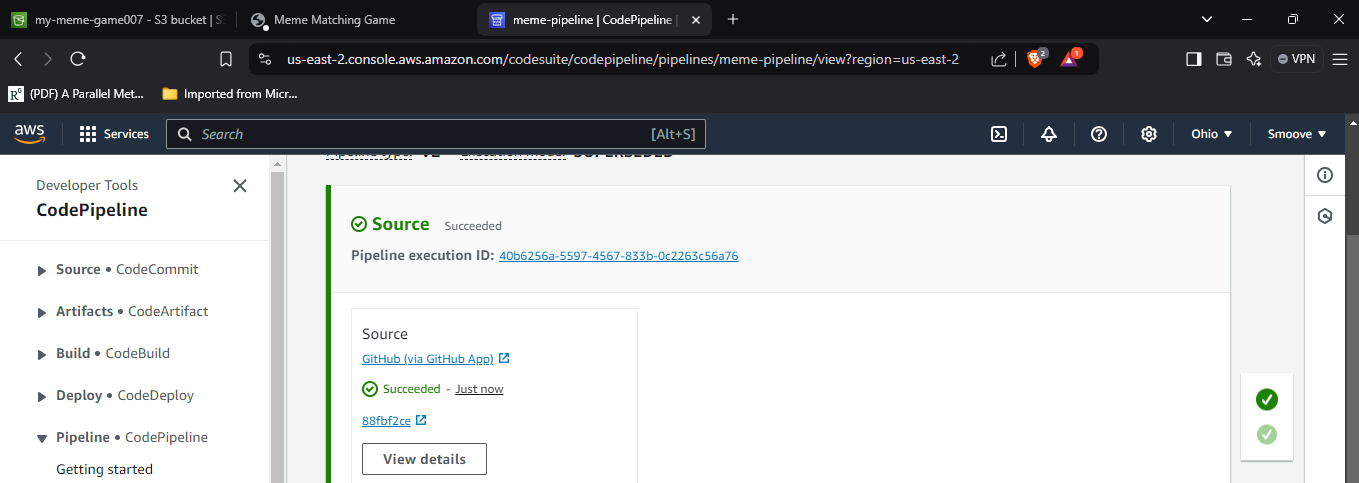
We are just going to add the word ‘UPDATED’ to the instructions section of the code, and then click on ‘Commit changes’ like this:



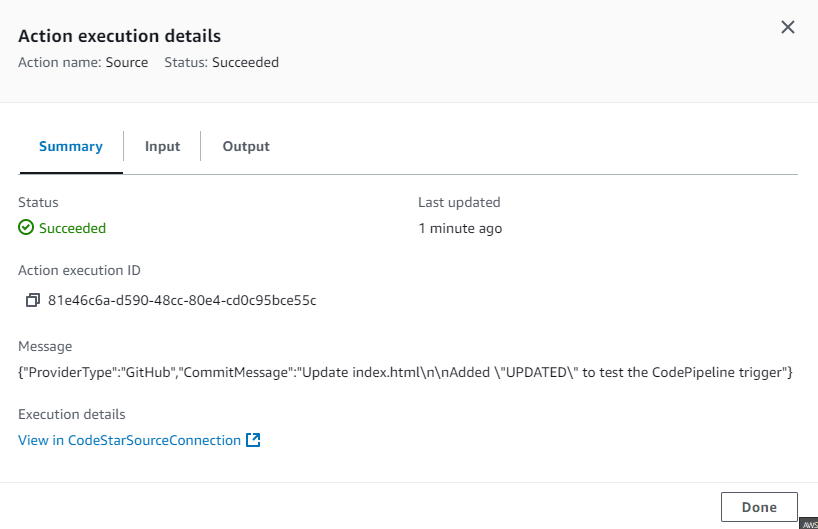
On the next page, you can just add a description of the change you just did, and then click on ‘Commit changes’.



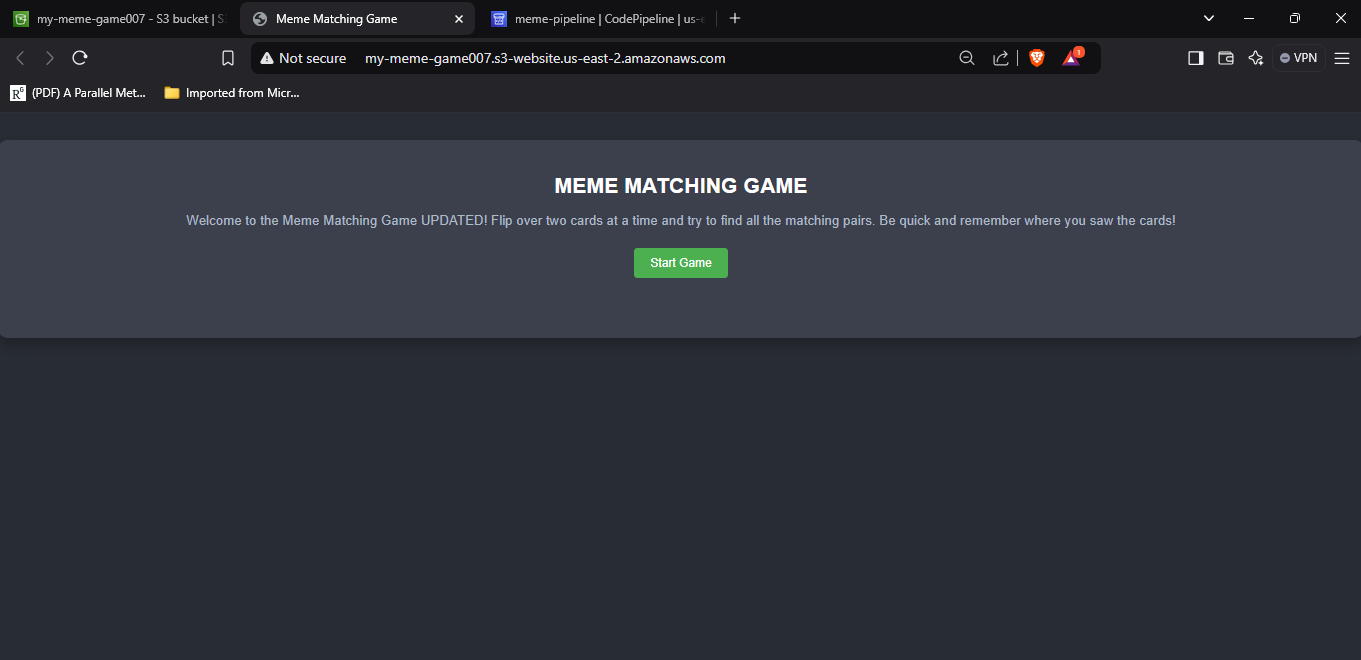
If you go back to the CodePipeline page, you will notice that some changes have occurred in the ‘Source’ section.



Click on ‘View details’ to show more information about this:

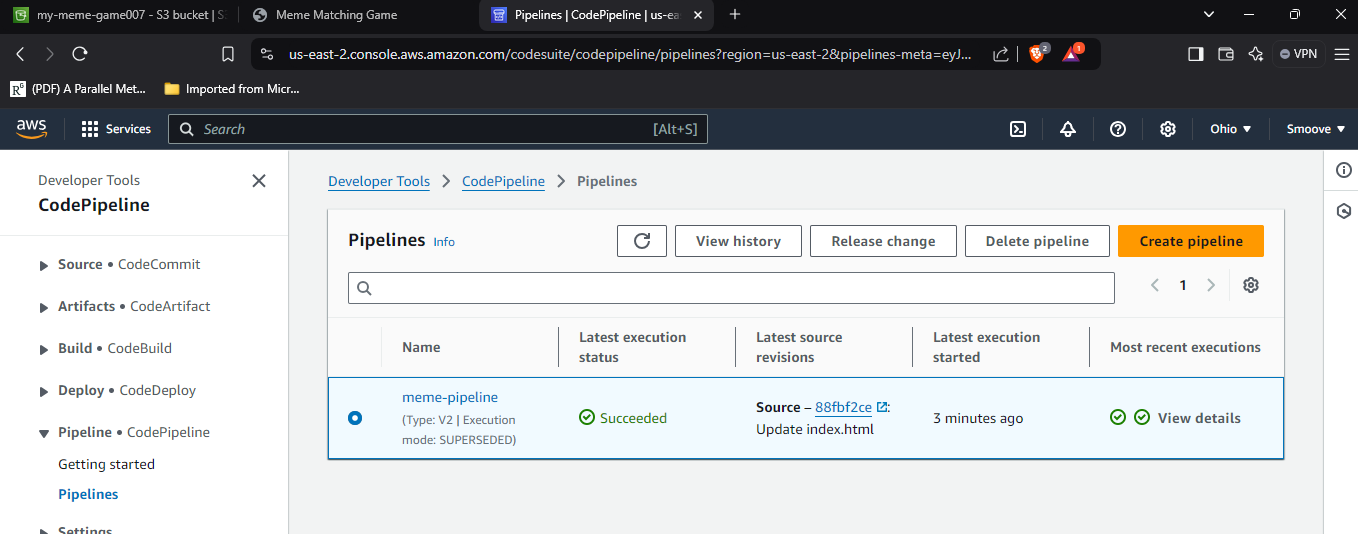


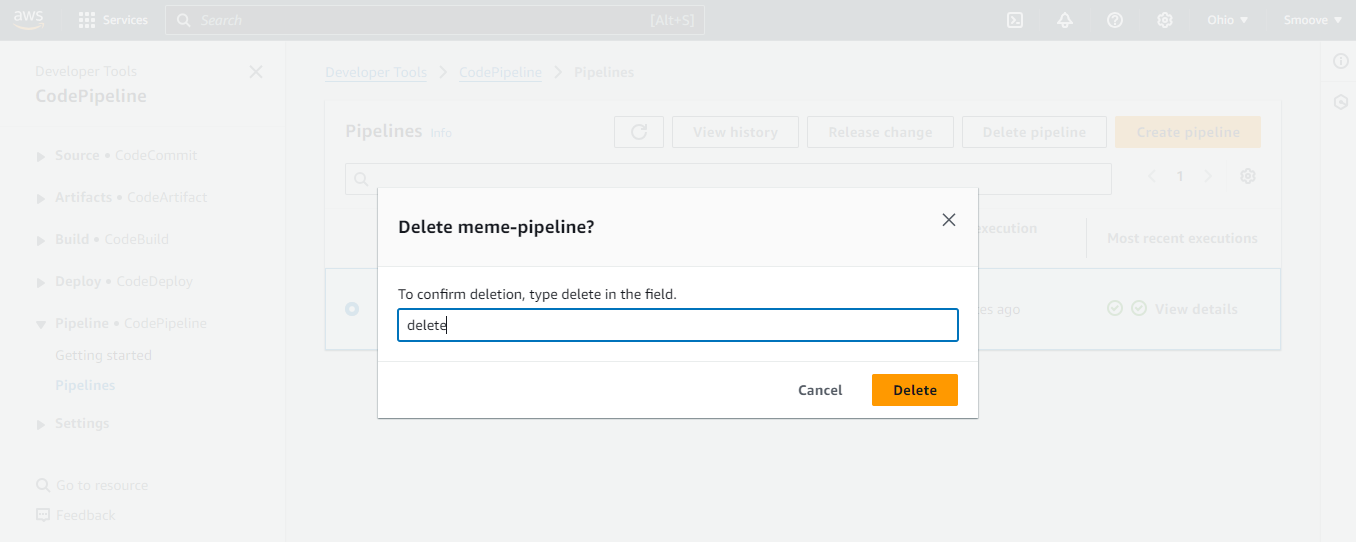
Now, go back to the webpage and reload. You will see that the little update we did has taken effect.



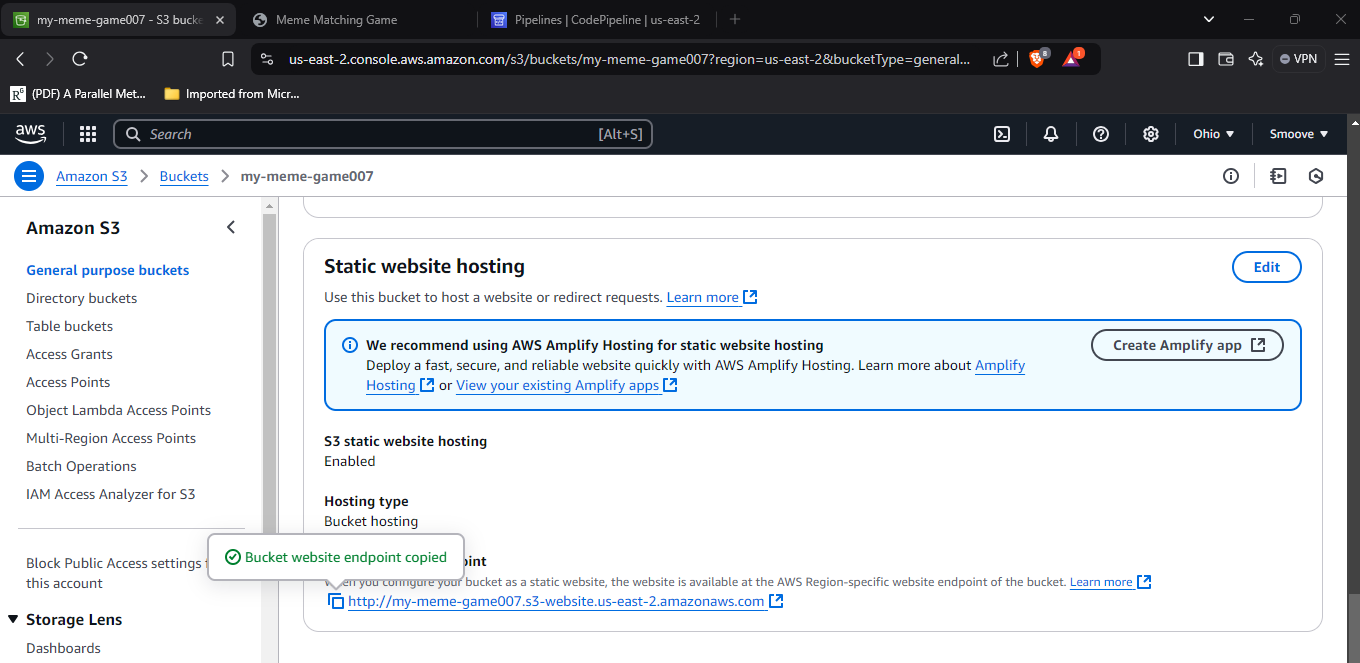
Great! So that’s all for this project. But remember to delete all your work to avoid incurring unwarranted charges!

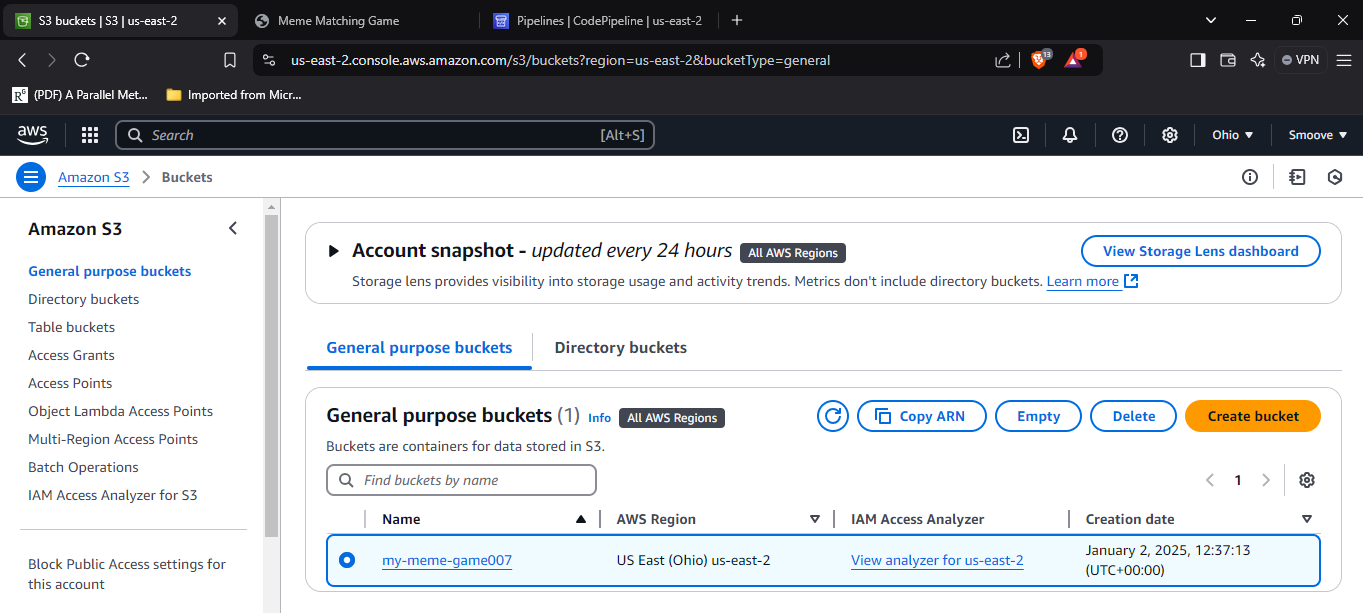
To do this, let’s begin with CodePipeline. Go to that page and select the pipeline you have created. Then, click on ‘Delete’ to remove it permanently.

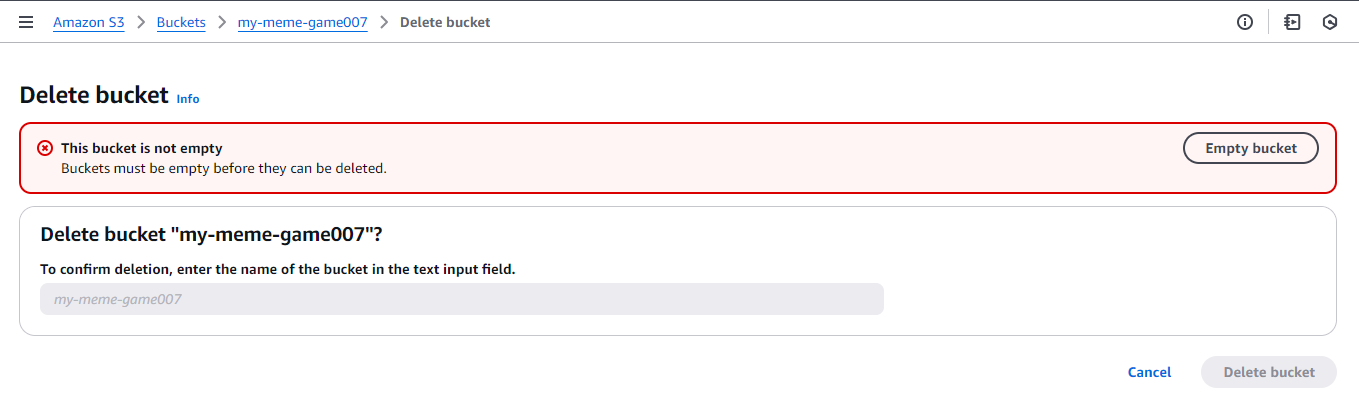




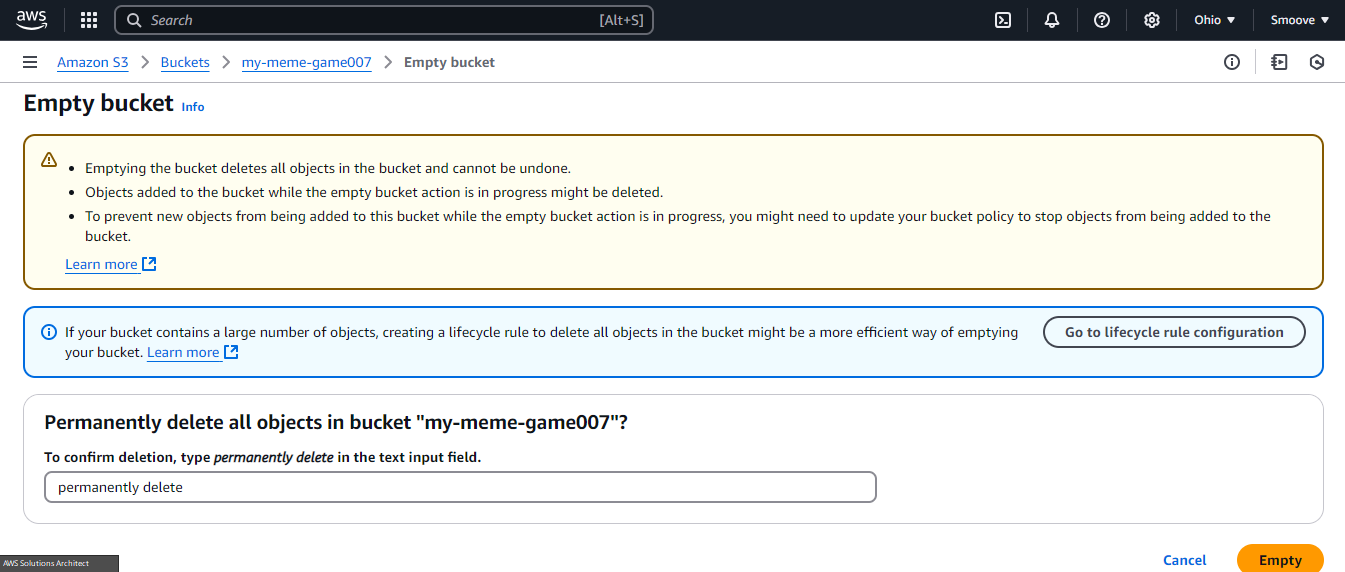
Now we move out to the S3 tab as well. Here we will try to delete the bucket, but since it has objects in it, we will have to empty it first. See the screenshots below.

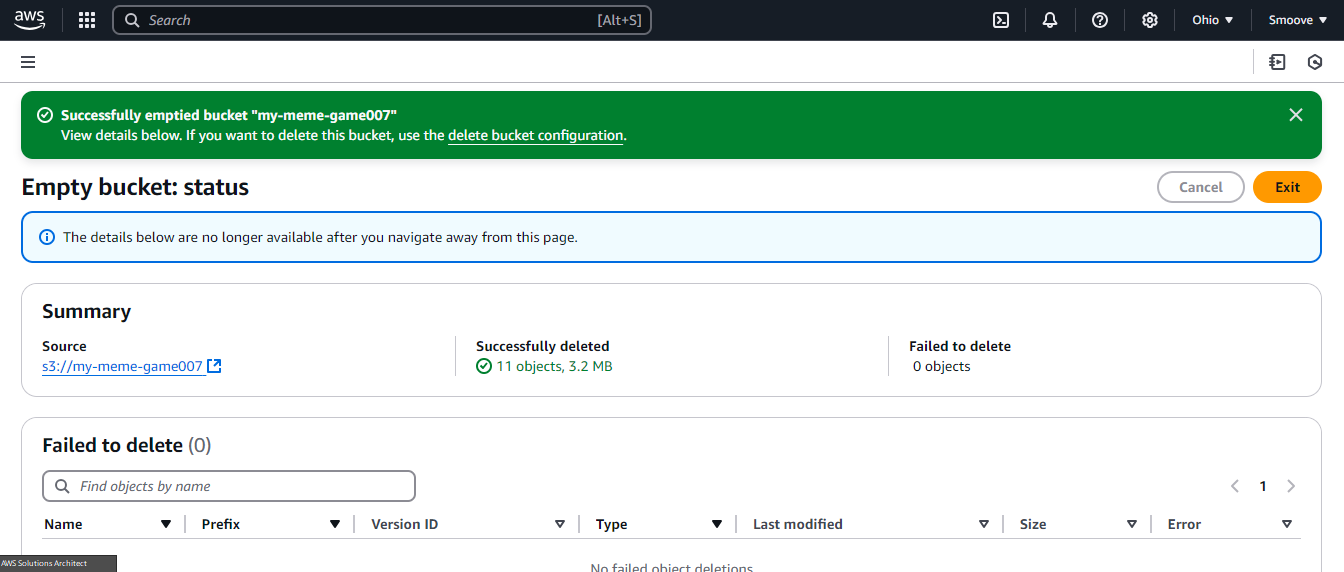




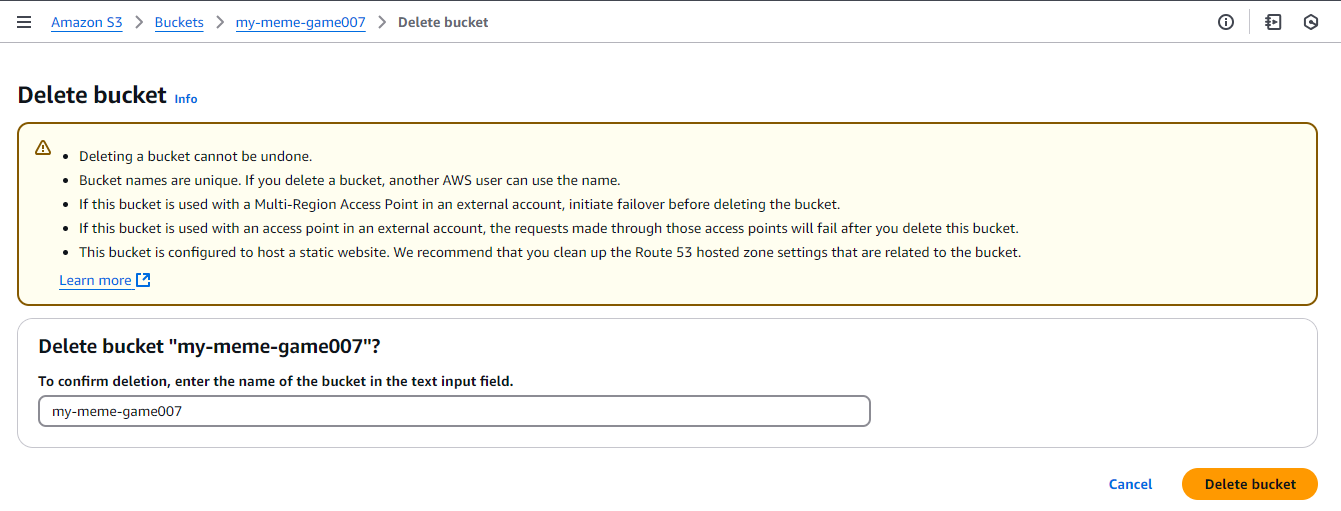


Type ‘permanently delete’ in the typing area, then click on ‘Empty’ to empty your bucket.





Now you can go ahead and delete the bucket. Click on ‘delete bucket configuration’ above, then go ahead and delete the bucket as shown below.



And we’re done!

I will be uploading more project tutorials as and when I go through them. Thanks!