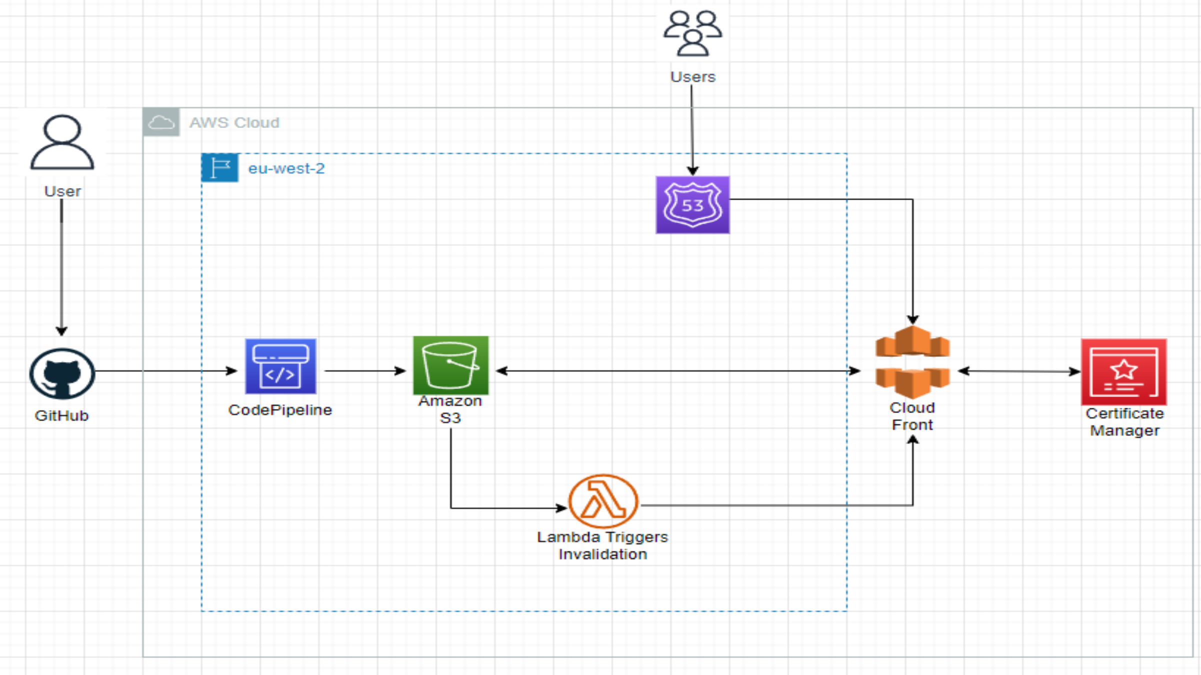
# Static Website hosted on S3 with CI/CD (step-by-step guide)

## Introduction:

This tutorial will guide you through the process of hosting a static website on Amazon S3 and making it accessible worldwide using Amazon CloudFront. You'll also learn how to secure your website using OAC to keep the bucket private, AWS Certificate Manager to protect and secure your website, and manage your domain with Amazon Route 53. Last but not least, we will use GitHub and AWS CodePipeline to automate every change we want to make to our website.

Static websites are a cost-effective and straightforward way to establish an online presence, whether it's for personal use, a blog, or a business. By following the steps in this guide, you can quickly host and manage your own static website with AWS's robust cloud services. This setup ensures speed, security, reliability, along with a custom domain and HTTPS support.

Here is an architecture on what you will be able to build by the end of this tutorial:



## Prerequisites:

**AWS Account**: To follow the examples and implement the project, you will need an AWS account.

**AWS Services**: While detailed knowledge of AWS services is not required, having a general awareness of services like Amazon S3, Amazon CloudFront, AWS CodePipeline, Lambda, AWS Identity and Access Management (IAM), AWS Certificate Manager, and Amazon Route 53 will be helpful. This tutorial will guide you through the setup and usage of these services.

**GitHub Account**: You should have a GitHub account for version control and collaborative development. Basic knowledge of Git for version control is a plus but not strictly required, as this tutorial may provide some Git basics.

**Visual Studio Code**: Familiarity with Visual Studio Code (VS Code) is recommended, including how to open, edit, and save files using this code editor.

**Web Development Basics** (Optional): Some experience with web development technologies (HTML, CSS, JavaScript) is helpful but not mandatory, as this tutorial will primarily focus on the AWS and DevOps aspects.

## <u>Audience:</u>

This tutorial is designed to accommodate a broad range of users, from beginners to advanced AWS practitioners. Whether you're new to AWS services or an experienced cloud engineer, you'll find valuable information and guidance tailored to your level of expertise.

**Beginners**: If you're just starting your AWS journey, this tutorial will provide step-by-step instructions and explanations to help you grasp the fundamentals and build a strong foundation in cloud static hosting.

**Intermediate** Users: If you have some experience with AWS but want to expand your knowledge and enhance your skills, you'll find in-depth insights, best practices, and tips for optimizing your projects.

**Advanced Users**: For experienced AWS professionals, this tutorial offers advanced configurations, automation strategies, and techniques to streamline your workflow and achieve a higher level of efficiency in deploying static websites.

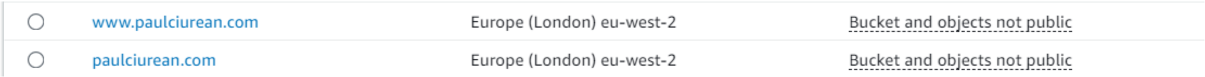
No matter where you fall on the spectrum of AWS expertise, this tutorial is designed to empower you with the knowledge and tools necessary to successfully host static websites and implement CI/CD pipelines in AWS.

## <u>PS</u>: You can find all the files you need for this tutorial on https://github.com/Paul-Ciurean/Tutorial-S3-with-CICD.git

# Step 1: Create our buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.
3. Enter the **Bucket name** for example, `example.com`.(main bucket)
4. Choose the Region where you want to create the bucket.

   Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.
5. To accept the default settings and create the bucket, choose **Create**.
6. Follow the above steps to create a 2nd bucket with the name www.example.com



| | | | |
|---|---|---|---|
| ○ | www.paulciurean.com | Europe (London) eu-west-2 | Bucket and objects not public |
| ○ | paulciurean.com | Europe (London) eu-west-2 | Bucket and objects not public |

We will keep our buckets private, as we will use Origin Access Control(OAC) to restrict users to access our buckets only through Cloud Front distribution.

To make sure that the users will be redirected to the main bucket, we will have to enable static website on our 2nd bucket.

1. We do that by going into our 2nd bucket, we chose **Properties,** and scroll all the way down to the bottom of the page.

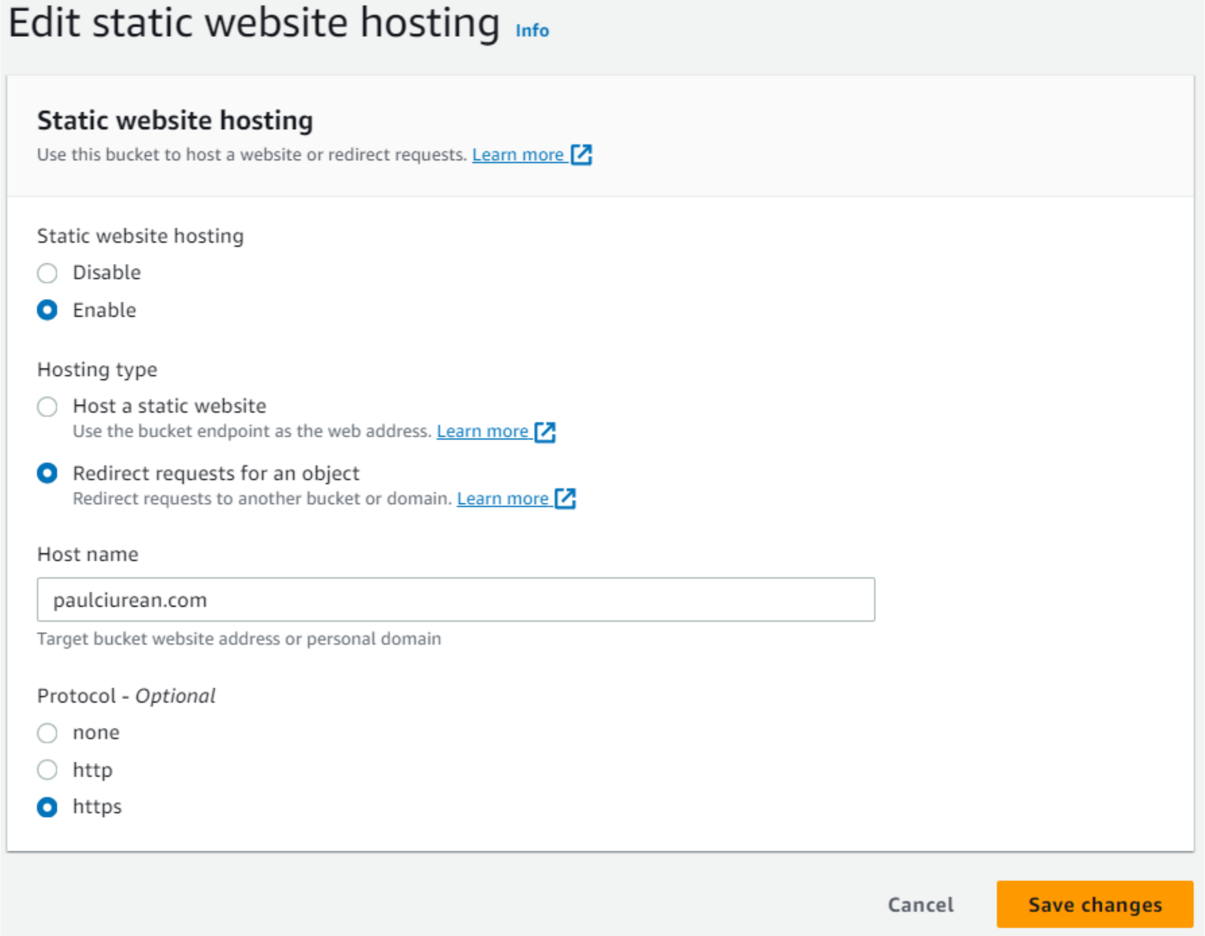
2. We click **Edit** and **Enable** for **Static website hosting.**

3. On the section **Hosting type** we chose **Redirect request for an object** and on the **Host name** we specify our main bucket.

4. On **Protocol** we select **HTTPS.**

5. **Save changes**.

Your page should look like this:



Edit static website hosting Info

**Static website hosting**

Use this bucket to host a website or redirect requests. Learn more

Static website hosting

○ Disable

● Enable

Hosting type

○ Host a static website
Use the bucket endpoint as the web address. Learn more

● Redirect requests for an object
Redirect requests to another bucket or domain. Learn more

Host name

paulciurean.com

Target bucket website address or personal domain

Protocol - *Optional*

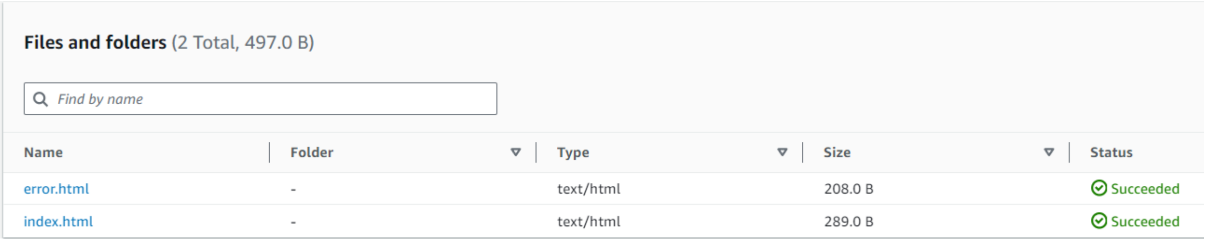
○ none

○ http

● https

## Uploading Website files:

You should upload your Website files in the main bucket( paulciurean.com). For the Static Website you should have an **index.html** file and **error.html**(optional) file. In my **index.html**



**Files and folders** (2 Total, 497.0 B)

| Name | Folder | Type | Size | Status |
|---|---|---|---|---|
| error.html | - | text/html | 208.0 B | ✓ Succeeded |
| index.html | - | text/html | 289.0 B | ✓ Succeeded |

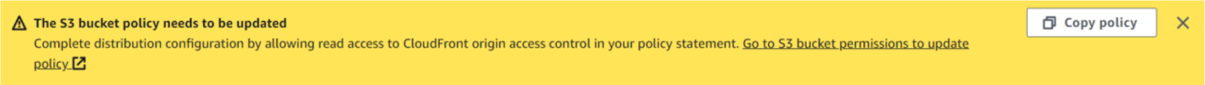
In my **index.html** I’ve got a basic webpage created in HTML.

## Step 2: Set up Cloud Front Distribution

1. In **Cloud Front** on **Distribution** page we **Create Distribution.**
2. On the **Origin page** for **Origin domain** we select our main S3 bucket.
3. We give a **Name** to our origin.
4. **Origin Access** we chose **Origin access control settings.**
5. **Origin access control** we create a control setting.
    - 5.1. In the new window we give a **Name.**
    - 5.2. **Origin type** we select **S3.**
    - 5.3. We keep the default option for the rest.
6. On the page **Default cache behaviour** at the section **Viewer protocol policy** we select **Redirect HTTP to HTTPS.** We keep the default option for the rest.
7. **Web Application Firewall:** We **Do not enable security protection.**
8. **Settings page:**
    - 8.1 **Alternate domain name -> Add item**: add the custom domain names. (**paulciurean.com** and **www.paulciurean.com**)
    - 8.2 **Custom SSL certificate:** we request a certificate.
    - 8.3 **Default root object: index.html** (name of our index file).

9. **Create distribution.**

When popped with the following message:



⚠ **The S3 bucket policy needs to be updated**

Complete distribution configuration by allowing read access to CloudFront origin access control in your policy statement. Go to S3 bucket permissions to update policy

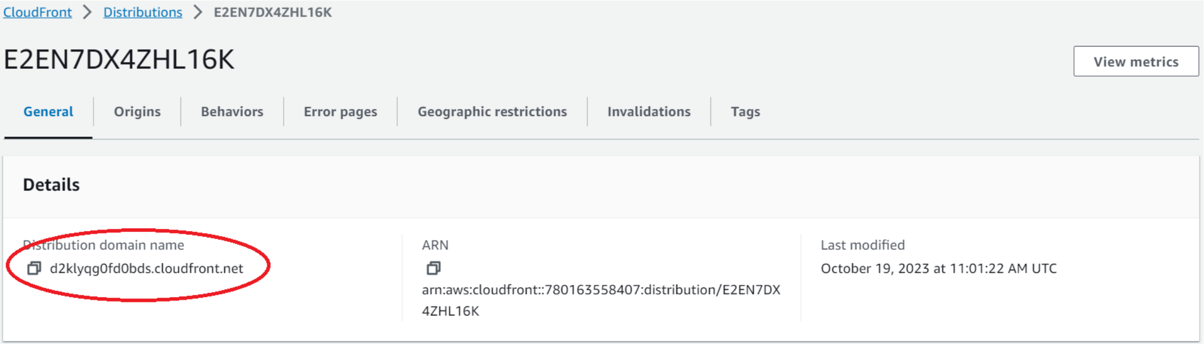
Copy the policy, we will have to complete our distribution configuration by allowing read access to CloudFront OAC in our S3 bucket policy statement.

To do that follow the next steps:

1. Go to our main bucket.
2. In **Permissions** scroll down to **Bucket policy** and click **Edit.**
3. Paste the policy you copied from the CloudFront distribution.

**PS: If you did not copy the policy after creating the distribution, you can find it in CloudFront -> Distribution -> Origins ->** select **Origins name** and click **Edit. On the following page you can find the policy.**

To test if our website is working, you can copy the **Distribution domain name** from your distribution:



E2EN7DX4ZHL16K

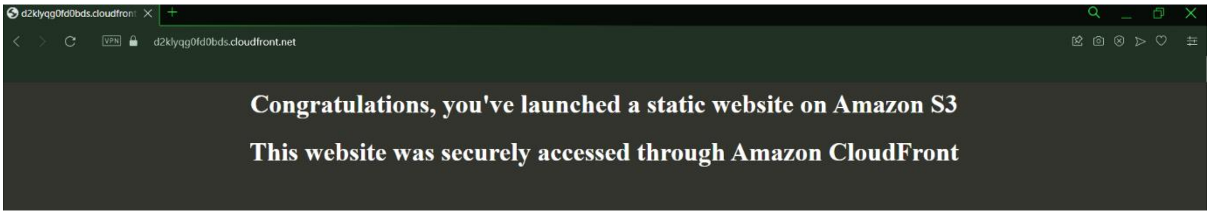
Details

Distribution domain name
d2klyqg0fd0bds.cloudfront.net

ARN
arn:aws:cloudfront::780163558407:distribution/E2EN7DX4ZHL16K

Last modified
October 19, 2023 at 11:01:22 AM UTC

And paste it in a new browser. My page looks like this:



Congratulations, you've launched a static website on Amazon S3

This website was securely accessed through Amazon CloudFront

**PS: If it doesn’t work that means the CloudFront distribution is not deployed yet. Give it few minutes and refresh the page.**

# Step 3: Domain and DNS Configuration

**Register a Domain** (if you don't already have one).

**Set up DNS Records**:

1. In **Route 53** on the **Hosted zone** page, **Create hosted zone.**
2. **Select the name of your website (paulciurean.com).**
3. **Type: Public hosted zone.**
4. **Create hosted zone.**

By default there will be 2 record names.

We now create 2 other records: 1 for paulciurean.com and 1 for www.paulciurean.com

5. Click **Create record.**
6. Click on **Switch to quick create.**
7. **Record name:** leave it blank. (when you create 2nd record you will have to type **www**).
8. For **Record type** chose **A.**
9. Above **Value** click **Alias**.
10. **Route traffic to:** Alias to CloudFront distribution.
11. Chose your distribution created in **Step 2.**
12. Click **Create records.**
13. Follow the previous steps to create a record for "www".

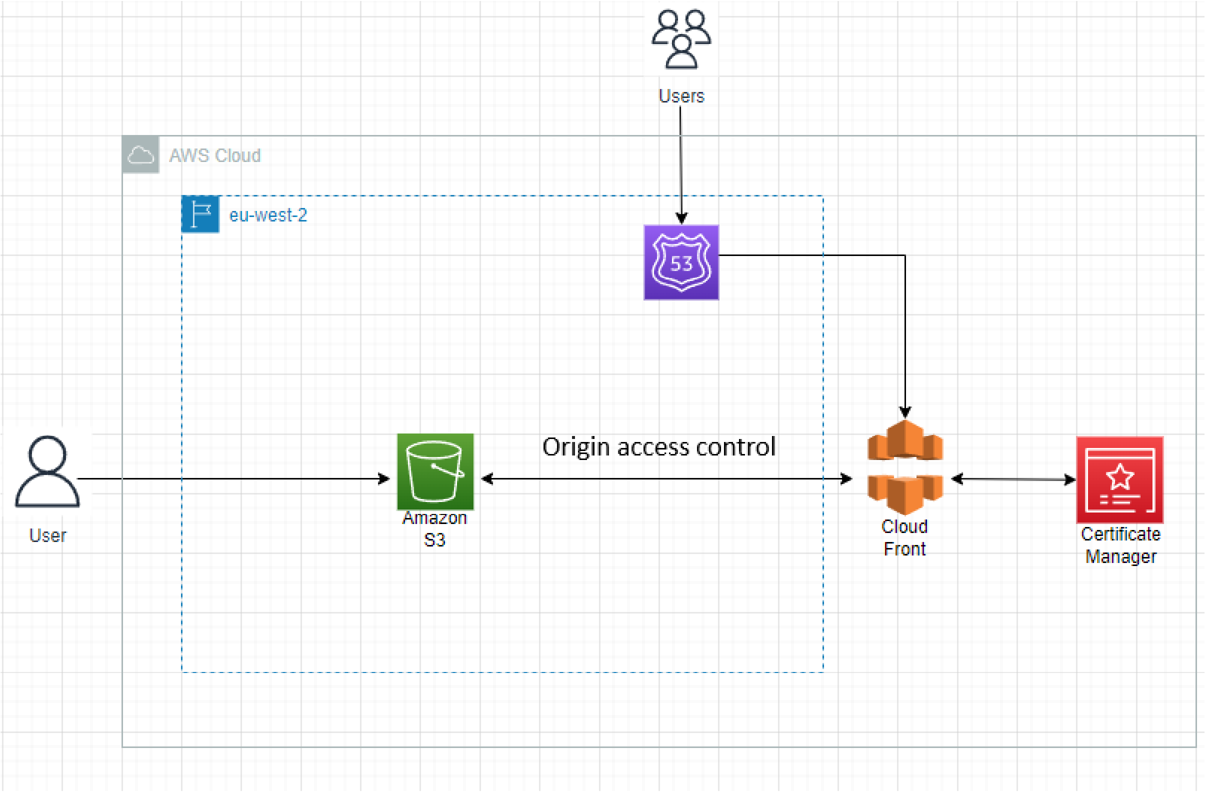
**You are now ready to test the website by opening a browser and browse the following URLs:**

http://your-domain-name, for example, example.com – Displays the index document in the your-domain-name bucket

http://www.your-domain-name for example, www.example.com – Redirects your request to the your-domain-name bucket

**PS: If it doesn't work that's because usually DNS changes will propagate within a few hours, but it can take up to 48 hours for everything to propagate across the Internet.**

## So far we achieved the following :



Up to this point, this is in detail tutorial on how to set up a static website using S3, keep the bucket private and secure using OAC, make it accessible worldwide using CloudFront for low latency, secure and protect your website with Certificate Manager, and manage your domain with Amazon Route 53.

If we want to make any changes to our website, we have to change the files (**index.html** and others such as **style.css** and **script.js** in case we want to add style to our website, not just keep it a blank page), delete and upload the files again in S3 manually, go to CloudFront and create an invalidation to remove the cache so the users can see the up-to-date site.

**Up next we will continue building a CI/CD pipeline to automate the process of making any changes to our website.**

## Step 4: Using Visual Studio Code and Github

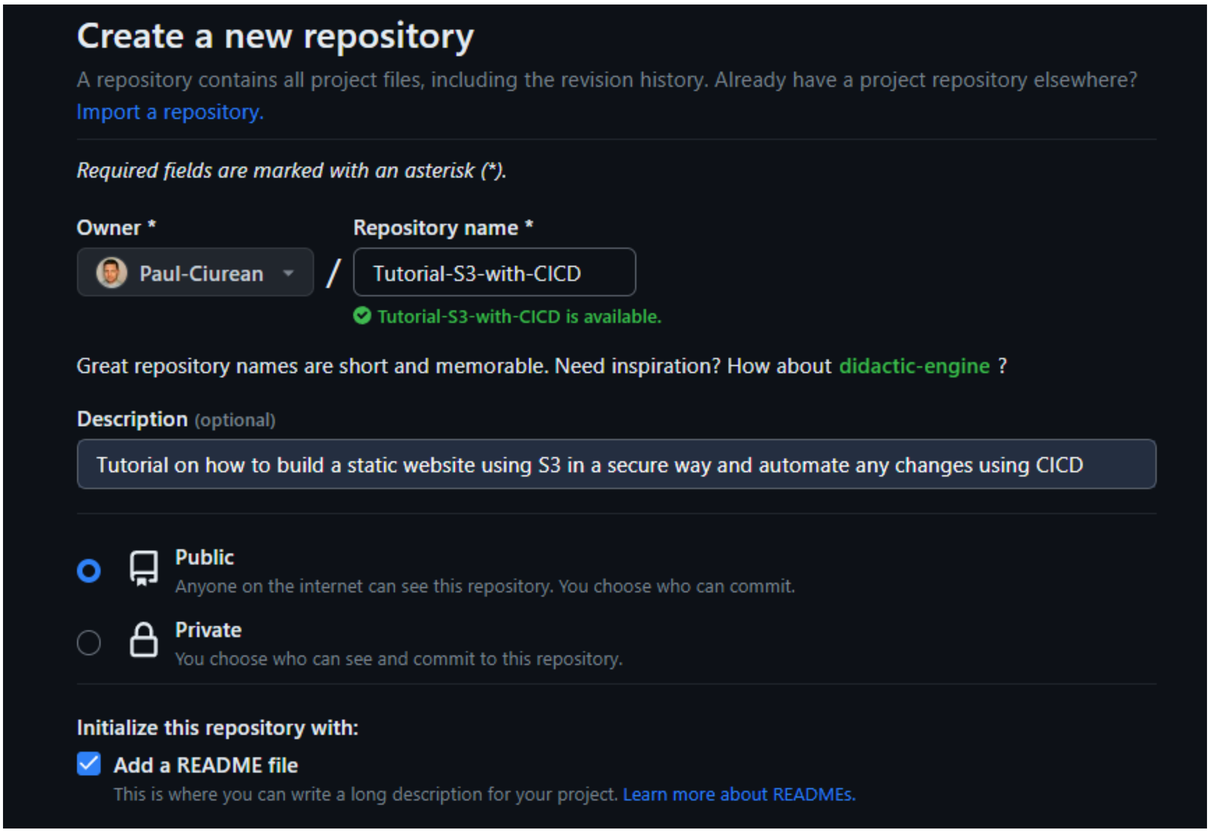
You should have a basic understanding on how to use VSC and github for the following part.

Here is a list of videos I watched to set up and get used to VSC and github:

https://www.youtube.com/watch?v=i_23KUAEtUM

https://www.youtube.com/watch?v=RGOj5yH7evk&t=1475s

1. Open your GitHub account and create a new repository.



### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

*Required fields are marked with an asterisk (*).*

Owner * Paul-Ciurean / Repository name * Tutorial-S3-with-CICD

Tutorial-S3-with-CICD is available.

Great repository names are short and memorable. Need inspiration? How about didactic-engine ?

Description (optional)

Tutorial on how to build a static website using S3 in a secure way and automate any changes using CICD

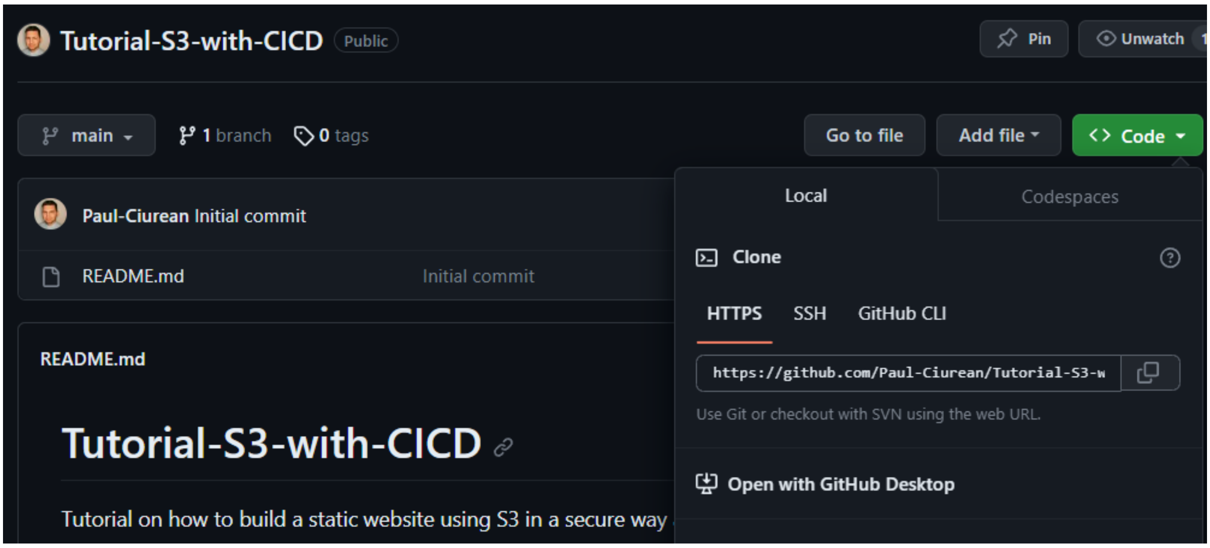
● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☑ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

2. Once the repository is created open VSC and press **Shift+CTRL+P**
3. Copy the link from **Local -> HTTPS** in your clipboard.

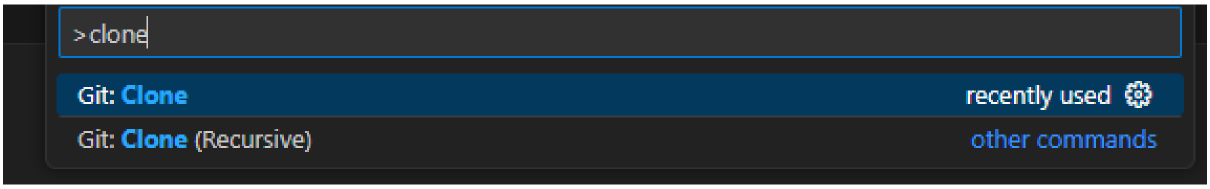


Tutorial on how to build a static website using S3 in a secure way

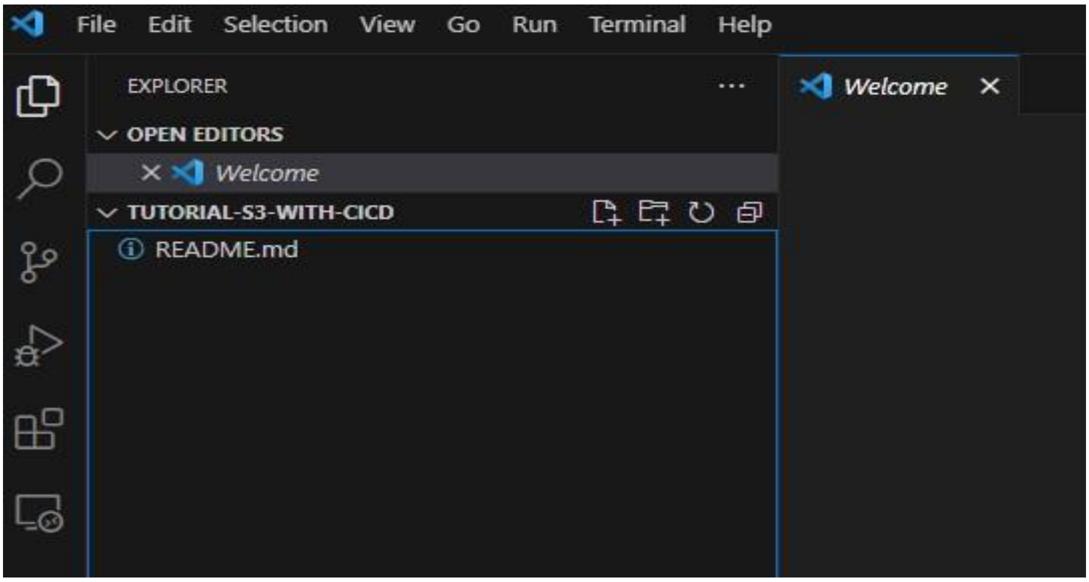
https://github.com/Paul-Ciurean/Tutorial-S3-w

Use Git or checkout with SVN using the web URL.

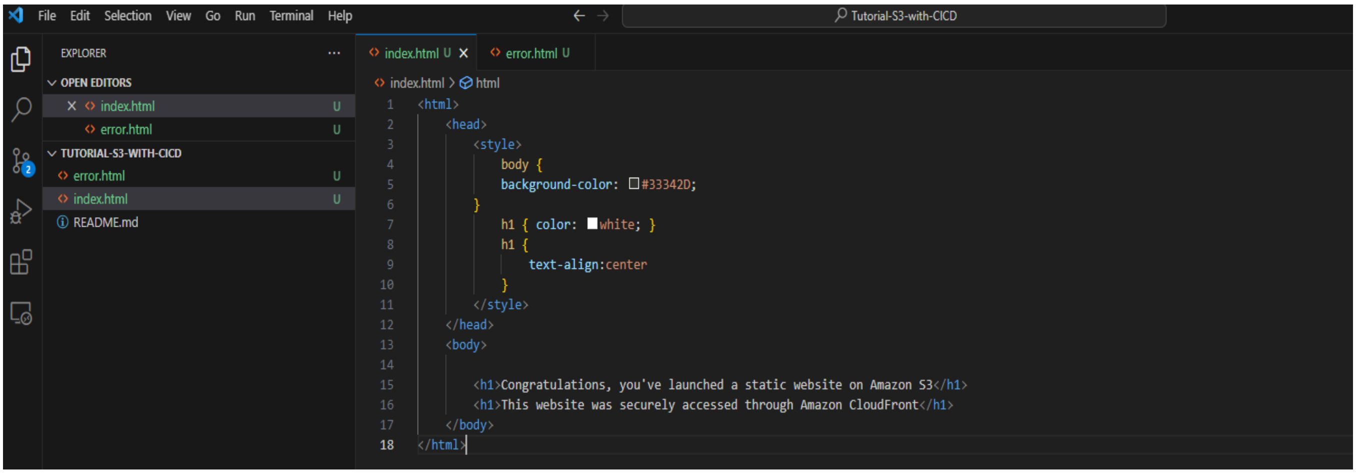
4. Type **clone** to clone the repository



5. Paste the link from Github website in the section above.
   That will take you to a window where you will have to select a folder where to keep your files locally on your machine.
   **PS: This will only work after you connect the github account with your local machine. Follow the tutorials from the youtube videos from above.**
6. You should now be in VSC with the repository open looking like this:



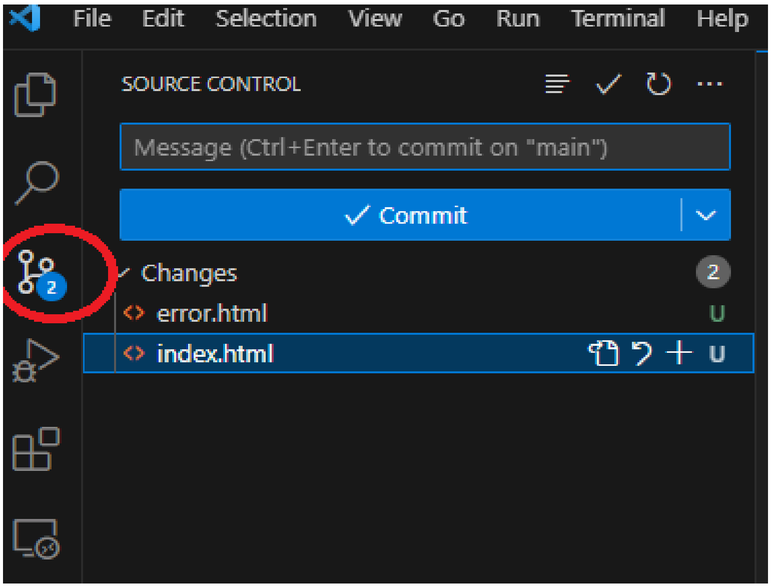
Here is the place where you can create your **index.html, styles.css,** and all the rest of the files for your website. In my repository you will be able to find the files I am using for this tutorial.



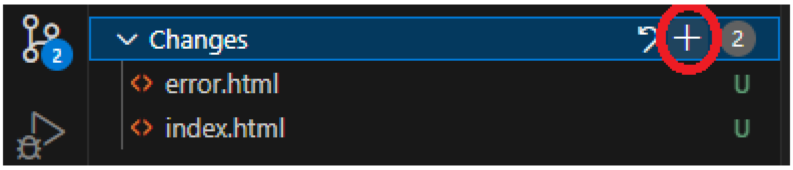
```
<html>
    <head>
        <style>
            body {
            background-color: □#33342D;
        }
            h1 { color: ■white; }
            h1 {
                text-align:center
            }
        </style>
    </head>
    <body>

        <h1>Congratulations, you've launched a static website on Amazon S3</h1>
        <h1>This website was securely accessed through Amazon CloudFront</h1>
    </body>
</html>
```

To commit the changes to the **github** repo into your online **github** account follow the next steps:

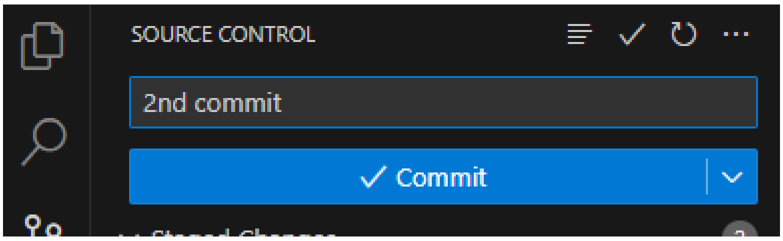
1. Click on source control on the left side of the page:



2. Stage all changes. Press on **+** button:



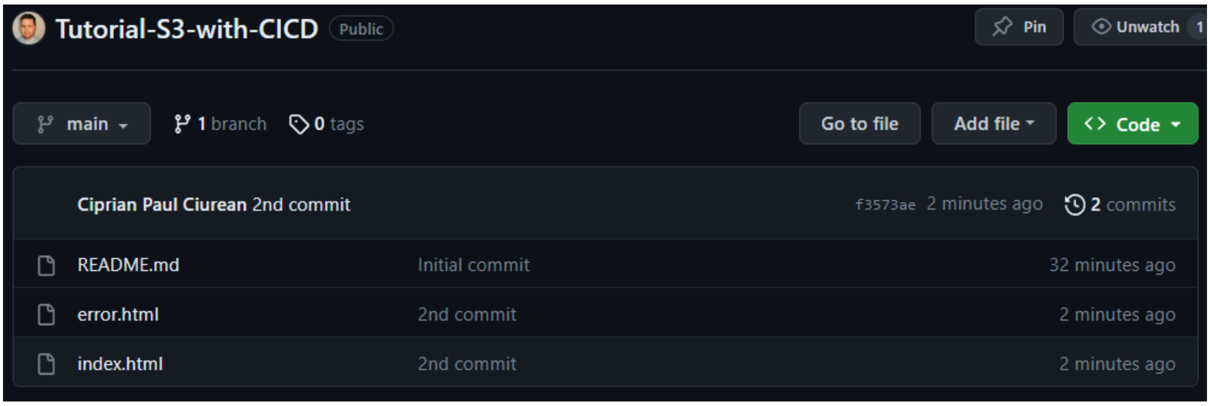
3. Type a message and press **Commit**.



2nd commit

4. Press **Sync Changes** and **OK** to commit the changes.

Now if you go online into your github account, you will find the 2 new files added to your repository.



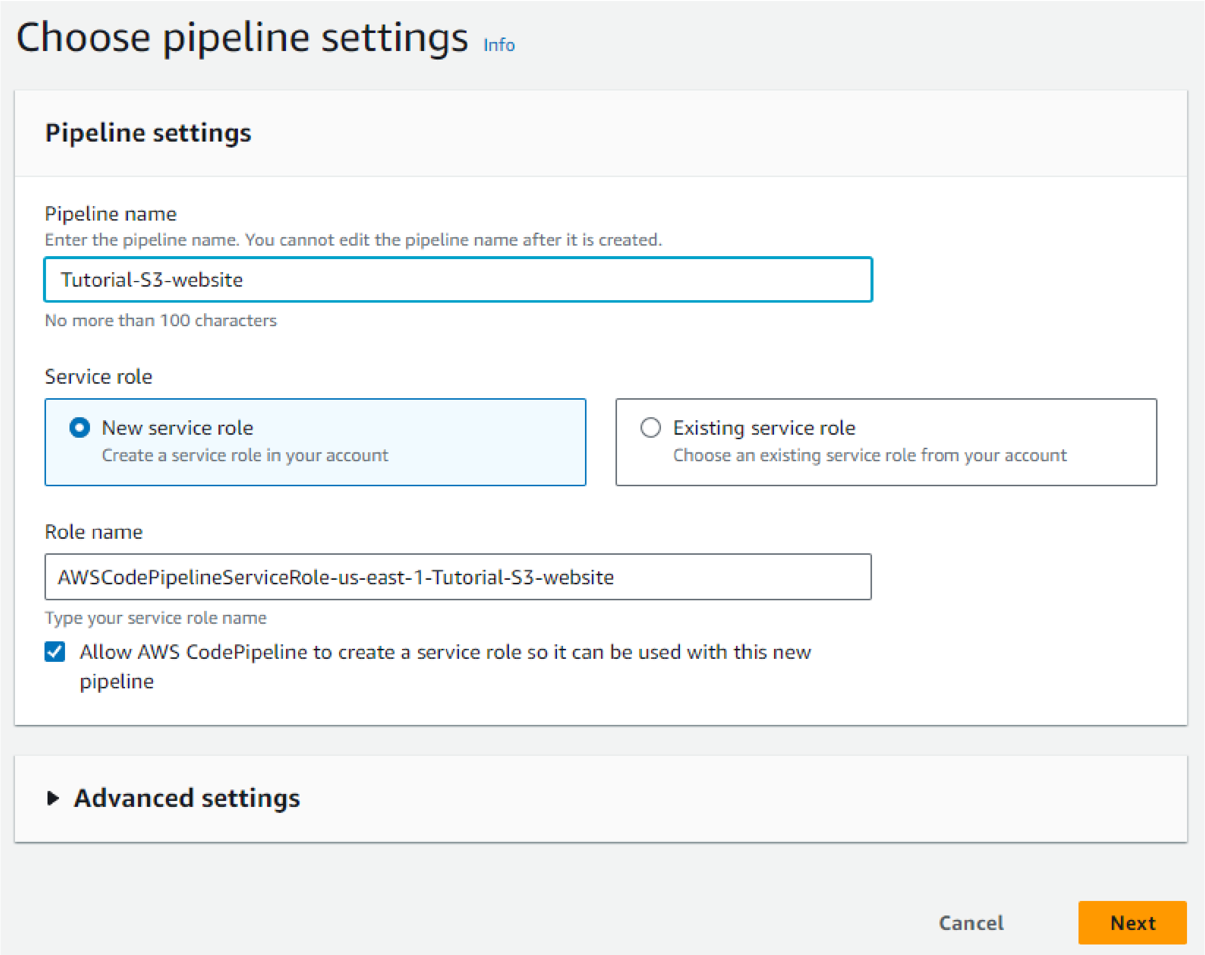
Ciprian Paul Ciurean 2nd commit f3573ae 2 minutes ago 2 commits

| | | |
|---|---|---|
| README.md | Initial commit | 32 minutes ago |
| error.html | 2nd commit | 2 minutes ago |
| index.html | 2nd commit | 2 minutes ago |

**Well done! We created our repository online where we will store our website files and this is where we will connect our pipeline to make the changes to our website easier.**

# Step 5: CI/CD Pipeline with AWS CodePipeline

## 5.1 Create the Pipeline:



Choose pipeline settings Info

**Pipeline settings**

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
Tutorial-S3-website
No more than 100 characters

Service role

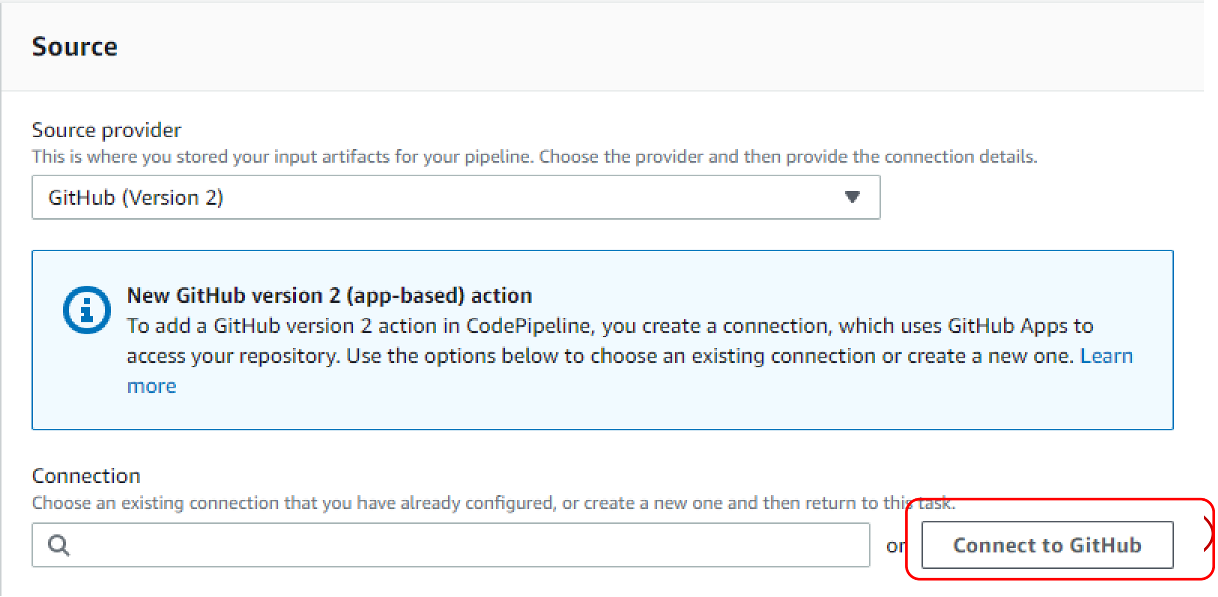
● New service role
Create a service role in your account

○ Existing service role
Choose an existing service role from your account

Role name
AWSCodePipelineServiceRole-us-east-1-Tutorial-S3-website
Type your service role name

☑ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

## 5.2 Add source stage:



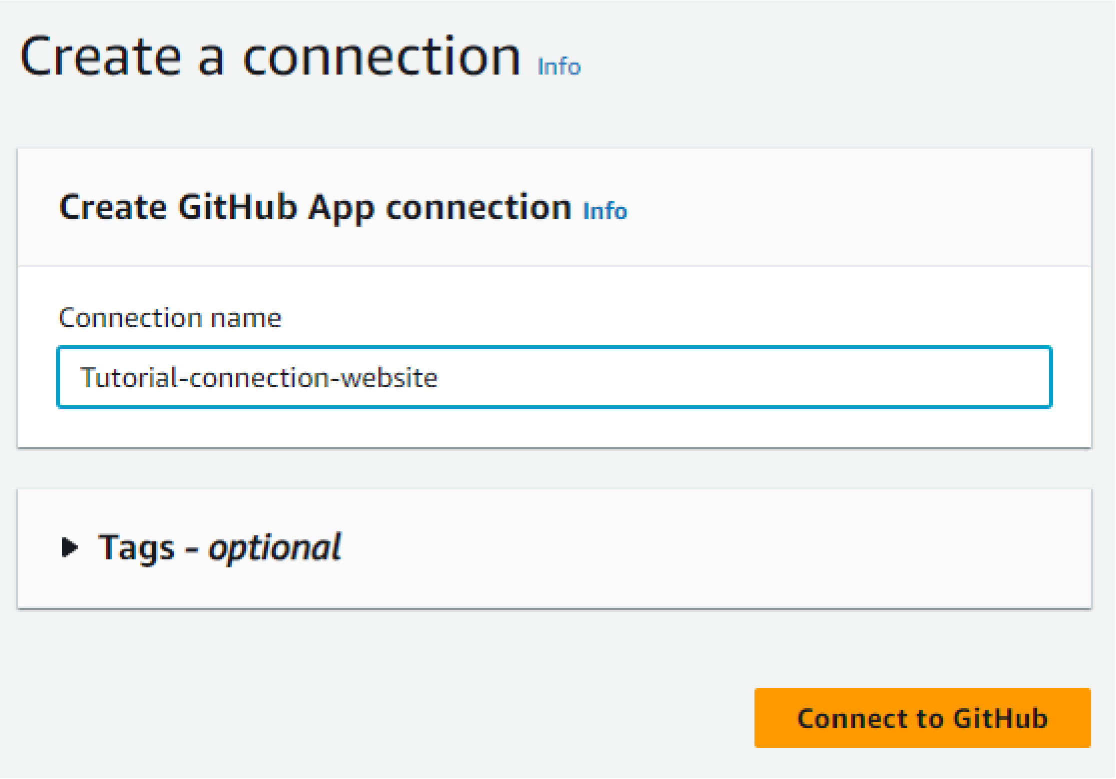
**Source**

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 2)

**New GitHub version 2 (app-based) action**
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. Learn more

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.

### 5.3 Create a connection:



Create a connection Info

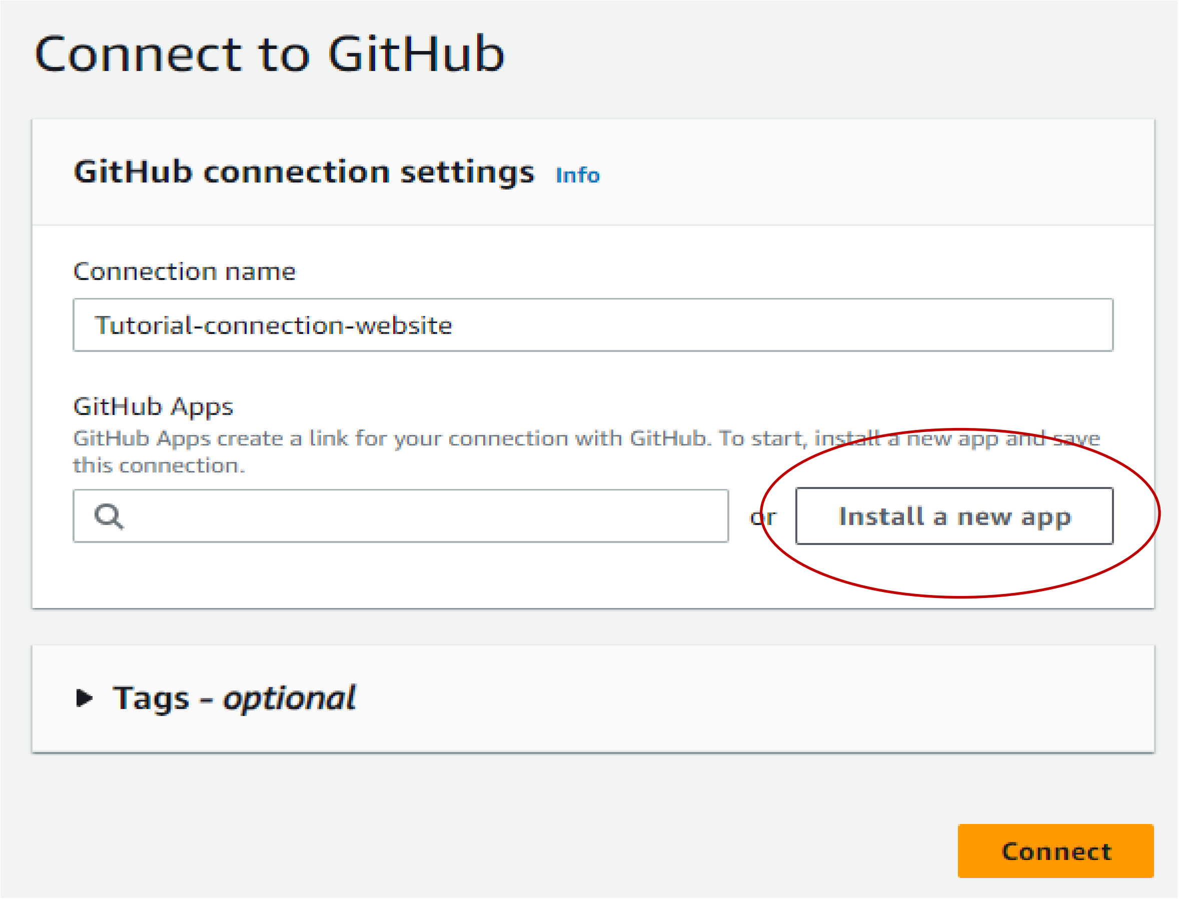
Create GitHub App connection Info

Connection name

Tutorial-connection-website

▶ Tags - *optional*

### 5.4 Connect to GitHub:



Connect to GitHub

GitHub connection settings Info

Connection name

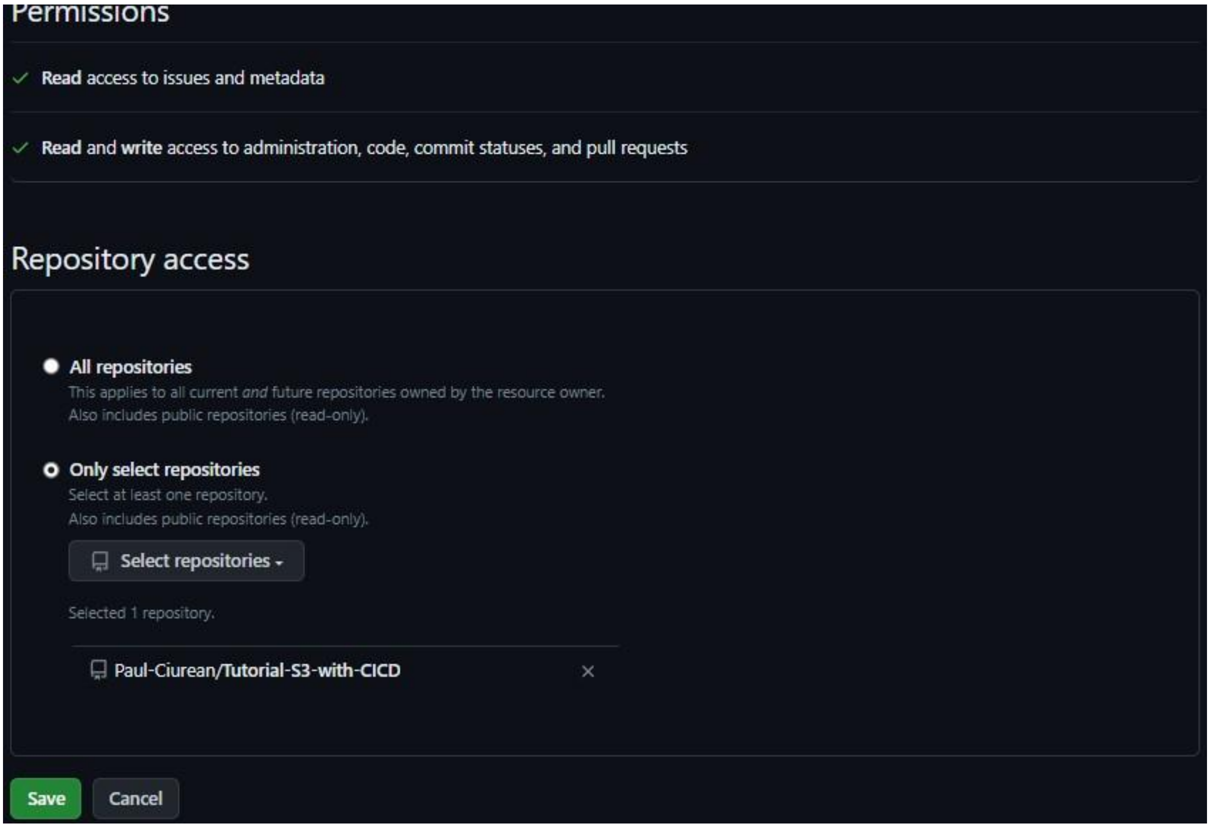
Tutorial-connection-website

GitHub Apps

GitHub Apps create a link for your connection with GitHub. To start, install a new app and save this connection.

▶ Tags - *optional*

## Install a new app:



Permissions

✓ Read access to issues and metadata

✓ Read and write access to administration, code, commit statuses, and pull requests

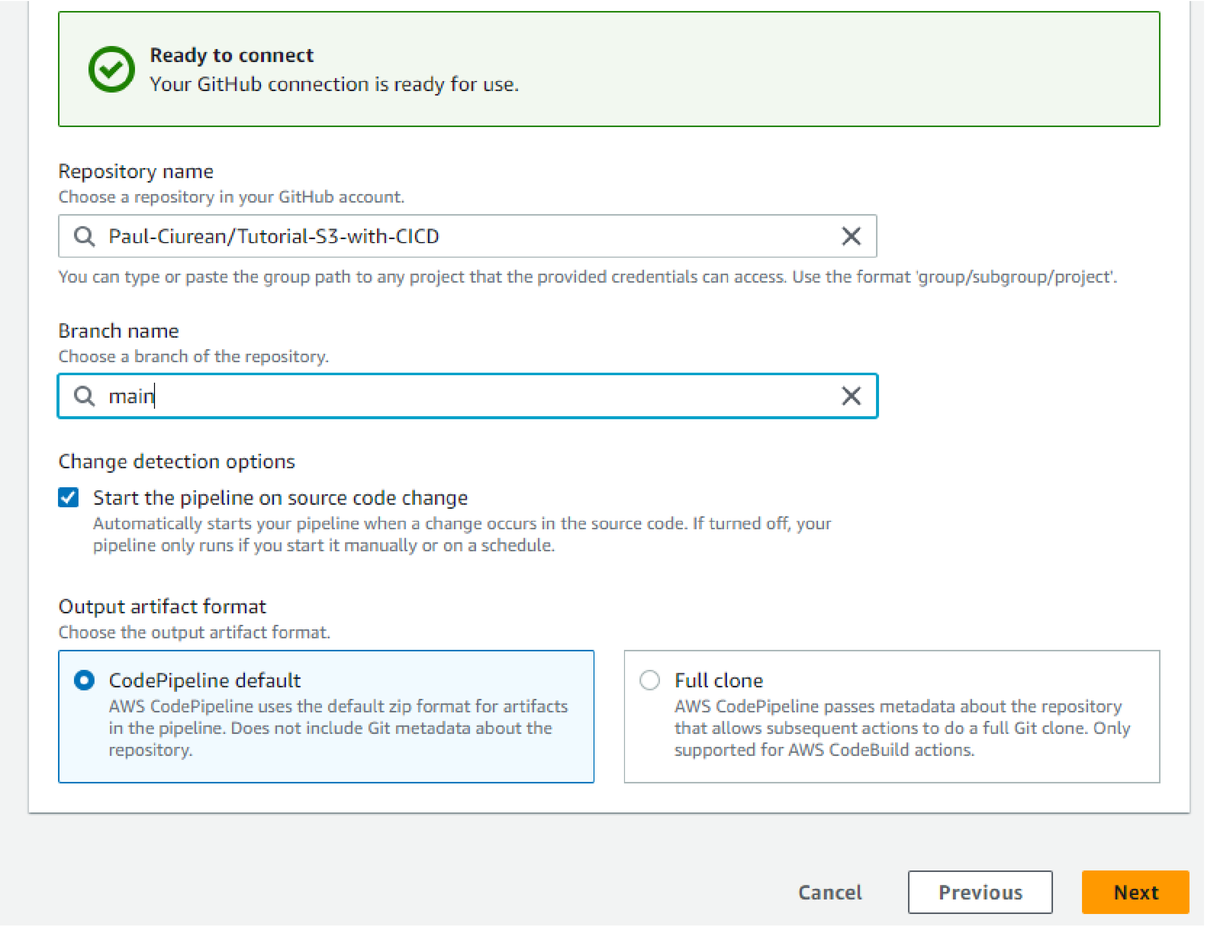
Repository access

○ **All repositories**
This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

● **Only select repositories**
Select at least one repository.
Also includes public repositories (read-only).

Selected 1 repository.

Paul-Ciurean/Tutorial-S3-with-CICD ×



**Ready to connect**
Your GitHub connection is ready for use.

Repository name
Choose a repository in your GitHub account.
Paul-Ciurean/Tutorial-S3-with-CICD
You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Branch name
Choose a branch of the repository.
main

Change detection options
☑ Start the pipeline on source code change
Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

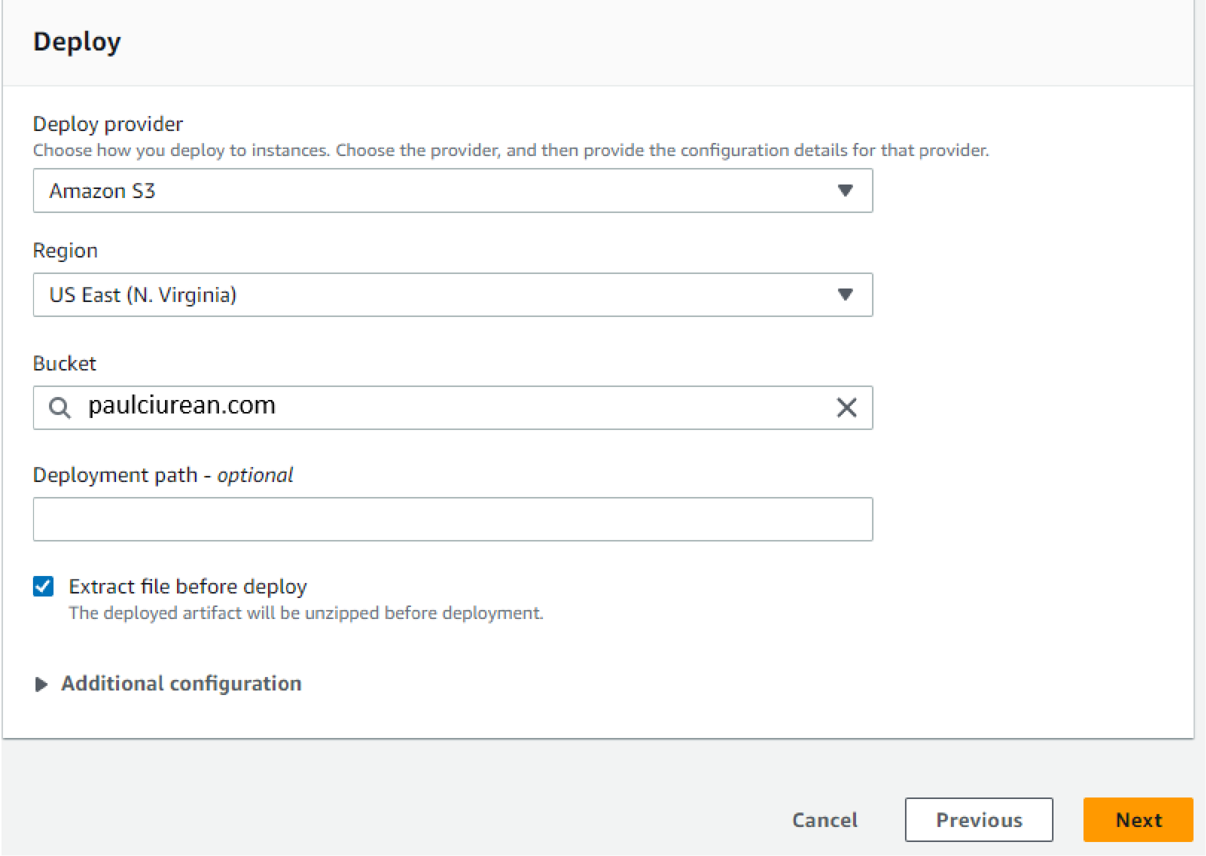
Output artifact format
Choose the output artifact format.

● CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

○ Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

**We can skip the building page as we build our files in VSC.**

## 5.5 Add deploy stage:



### Deploy

Deploy provider

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon S3

Region

US East (N. Virginia)

Bucket

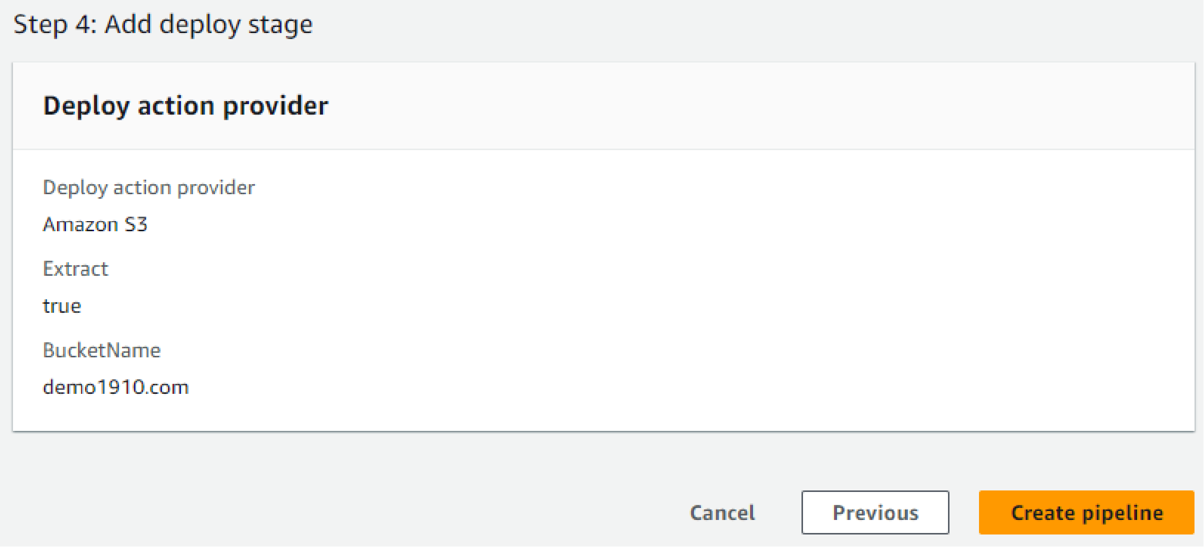
paulciurean.com

Deployment path - *optional*

☑ Extract file before deploy

The deployed artifact will be unzipped before deployment.

## 5.6 Review the pipeline:



Step 4: Add deploy stage

### Deploy action provider

Deploy action provider

Amazon S3

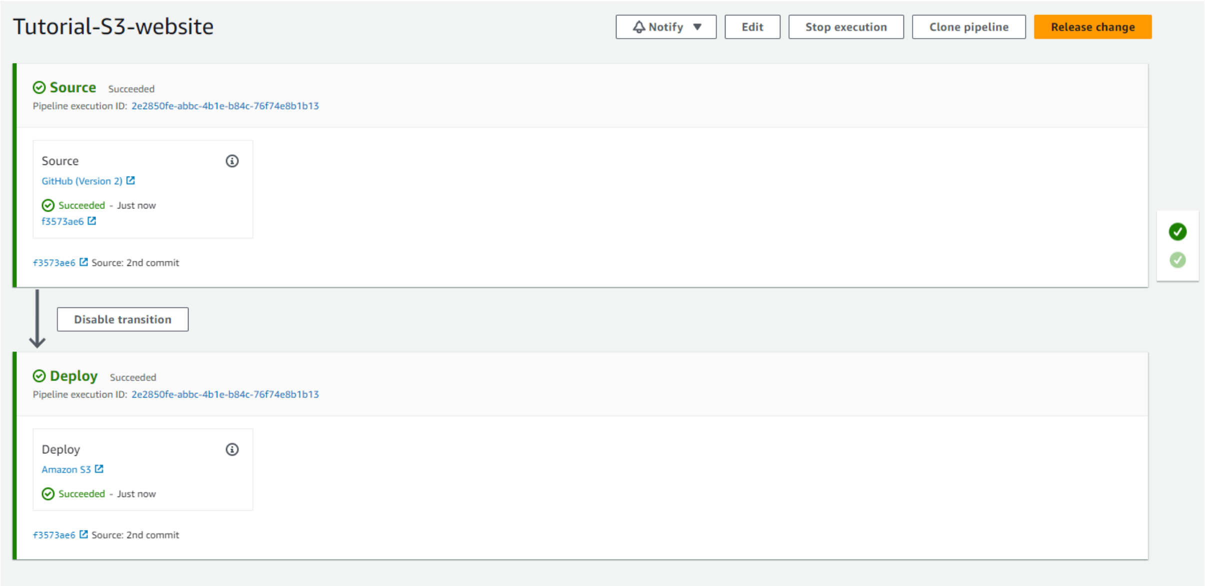
Extract

true

BucketName

demo1910.com

## AWS test the pipeline and we receive a SUCCESS



Tutorial-S3-website

Source Succeeded
Pipeline execution ID: 2e2850fe-abbc-4b1e-b84c-76f74e8b1b13

Source
GitHub (Version 2)
Succeeded - Just now
f3573ae6

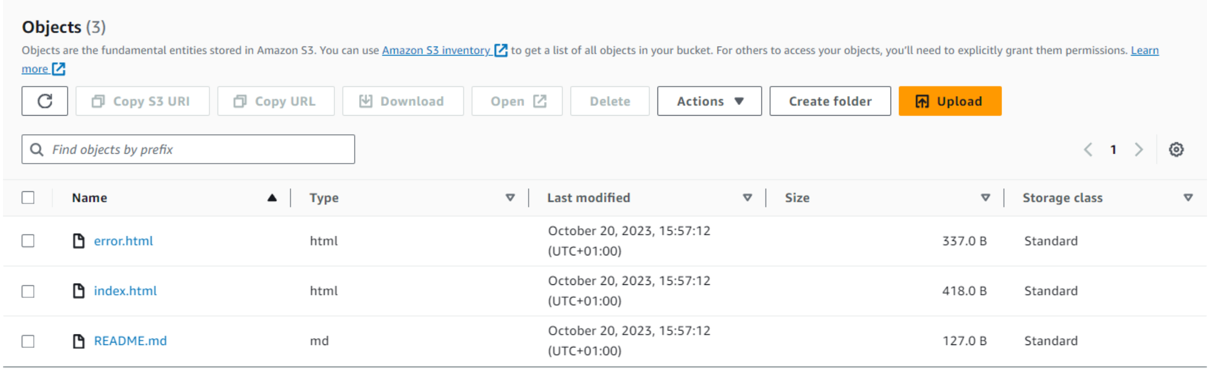
f3573ae6 Source: 2nd commit

Deploy Succeeded
Pipeline execution ID: 2e2850fe-abbc-4b1e-b84c-76f74e8b1b13

Deploy
Amazon S3
Succeeded - Just now

f3573ae6 Source: 2nd commit

CodePipeline is now pulling any changes from GitHub and deploying them to the bucket that the website is hosted in. Looking at the S3 bucket you can see all the files have been updated at the same time. This is because CodePipeline deploys the entire repository rather than just individual changed files.



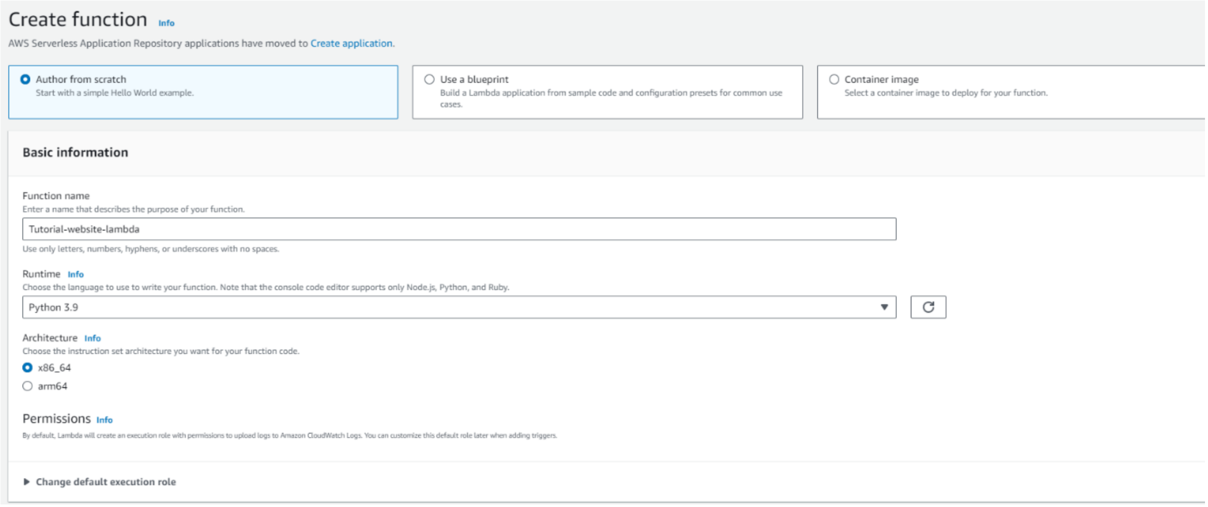
Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| □ | Name ▲ | Type | Last modified | Size | Storage class |
|---|---|---|---|---|---|
| □ | error.html | html | October 20, 2023, 15:57:12 (UTC+01:00) | 337.0 B | Standard |
| □ | index.html | html | October 20, 2023, 15:57:12 (UTC+01:00) | 418.0 B | Standard |
| □ | README.md | md | October 20, 2023, 15:57:12 (UTC+01:00) | 127.0 B | Standard |

# Step 6: Create the Lambda Function

## 6.1 Create Function:



Create function Info

AWS Serverless Application Repository applications have moved to Create application.

● Author from scratch
Start with a simple Hello World example.

○ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

○ Container image
Select a container image to deploy for your function.

**Basic information**

Function name
Enter a name that describes the purpose of your function.
Tutorial-website-lambda
Use only letters, numbers, hyphens, or underscores with no spaces.

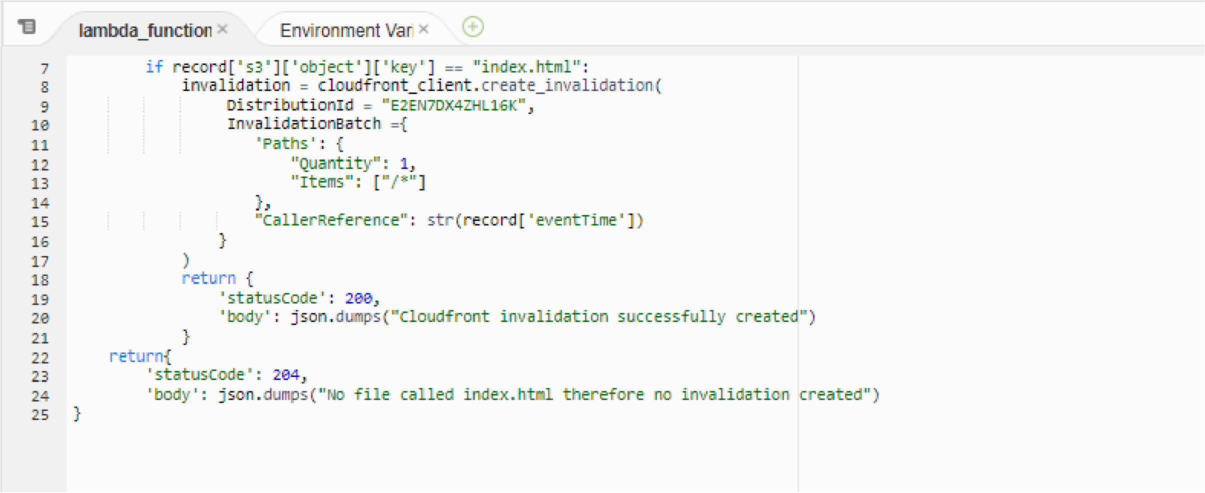
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
● x86_64
○ arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

## 6.2 Copy the python code in lambda function

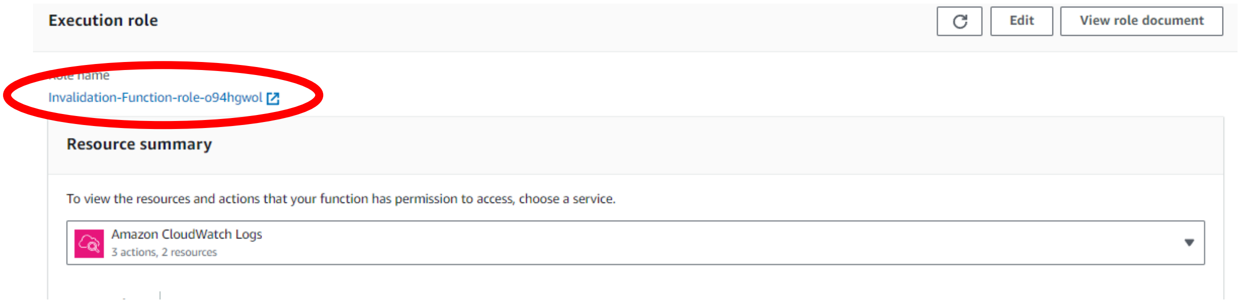


```
7          if record['s3']['object']['key'] == "index.html":
8              invalidation = cloudfront_client.create_invalidation(
9                  DistributionId = "E2EN7DX4ZHL16K",
10                 InvalidationBatch ={
11                     'Paths': {
12                         "Quantity": 1,
13                         "Items": ["/*"]
14                     },
15                     "CallerReference": str(record['eventTime'])
16                 }
17             )
18             return {
19                 'statusCode': 200,
20                 'body': json.dumps("Cloudfront invalidation successfully created")
21             }
22     return{
23         'statusCode': 204,
24         'body': json.dumps("No file called index.html therefore no invalidation created")
25 }
```

**PS: You can find the code in my GitHub but remember to insert your CloudFront distribution on line 5.**

## 6.2 Add permissions to Lambda to be able to create invalidation

### 6.2.1 In Lambda go to Configuration -> Permission



**Execution role**

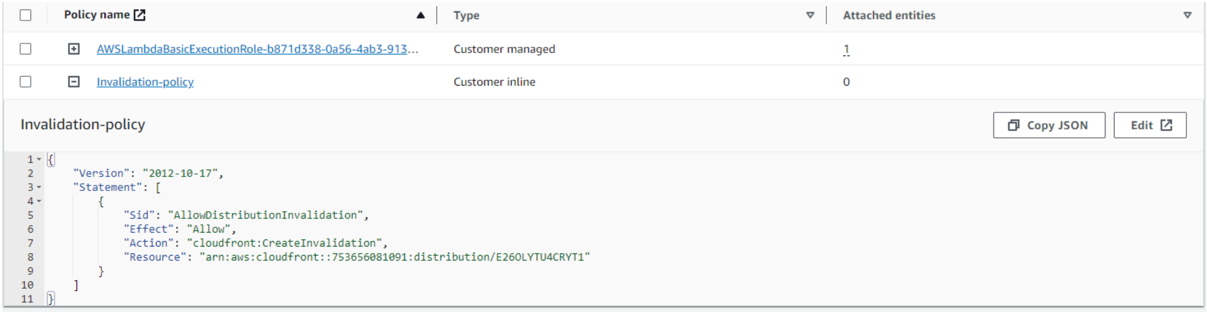
Role name
Invalidation-Function-role-o94hgwol

**Resource summary**

To view the resources and actions that your function has permission to access, choose a service.

Amazon CloudWatch Logs
3 actions, 2 resources

### 6.2.2 Add permissions



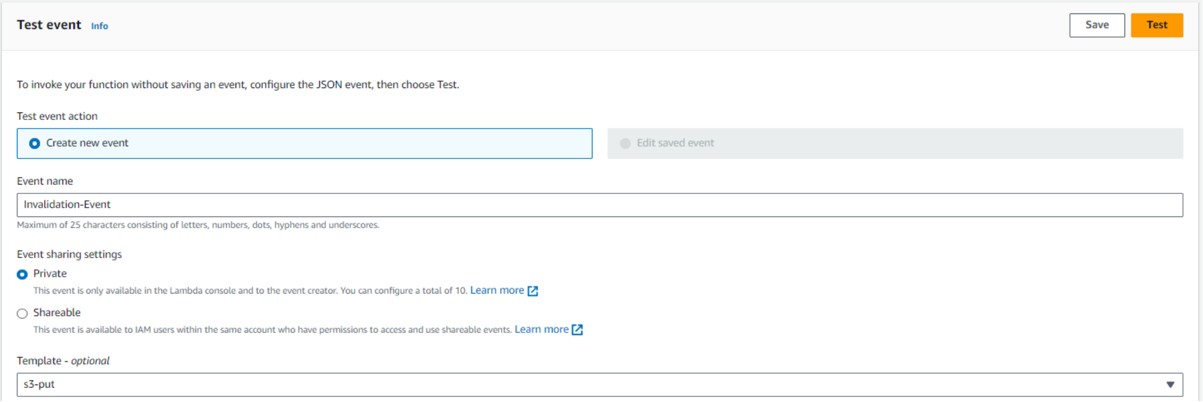
| □ | Policy name | Type | Attached entities |
|---|---|---|---|
| □ | AWSLambdaBasicExecutionRole-b871d338-0a56-4ab3-913… | Customer managed | 1 |
| □ | Invalidation-policy | Customer inline | 0 |

Invalidation-policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDistributionInvalidation",
            "Effect": "Allow",
            "Action": "cloudfront:CreateInvalidation",
            "Resource": "arn:aws:cloudfront::753656081091:distribution/E26OLYTU4CRYT1"
        }
    ]
}
```

**Create a new  inline policy. You can find the JSON for it in GitHub**

### 6.3 Create a test event



**Test event**

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

● Create new event    ○ Edit saved event

Event name

Invalidation-Event

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

● Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

○ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*

s3-put

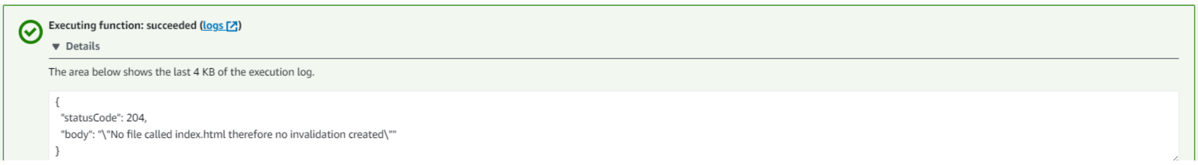
**Make sure you change the following:**

**Line 10 - replace “example” with your account number**

**Line 25 - replace “example” with your account number**

**Line 27 – replace “example-bucket” with your bucket name**

### 6.3 Test the function



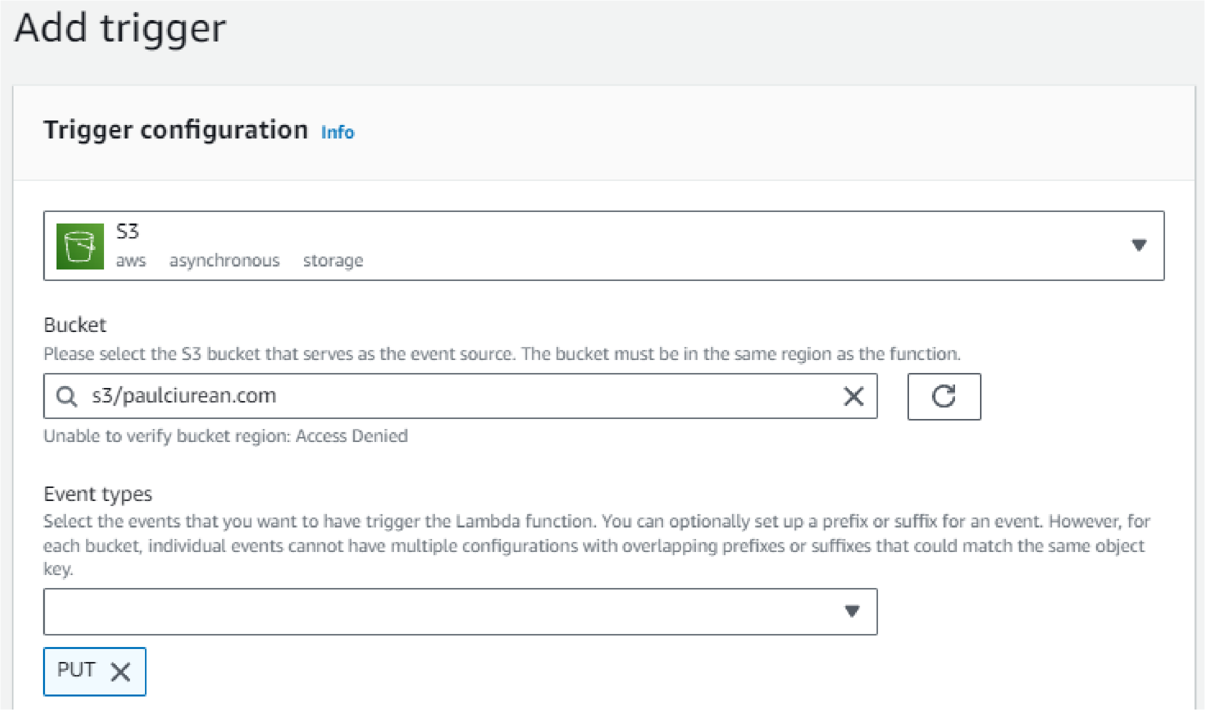
Executing function: succeeded (logs)

▼ Details

The area below shows the last 4 KB of the execution log.

```
{
 "statusCode": 204,
 "body": "\"No file called index.html therefore no invalidation created\""
}
```

### 6.4 Trigger lambda with S3 events



Add trigger

Trigger configuration Info

S3
aws asynchronous storage

Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

s3/paulciurean.com

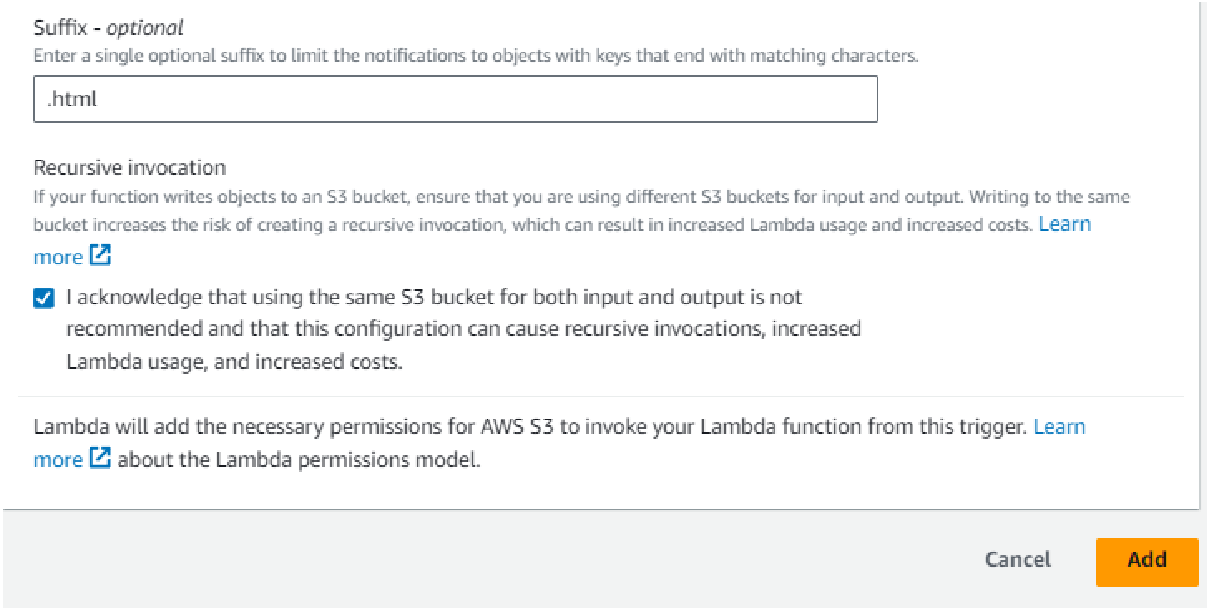
Unable to verify bucket region: Access Denied

Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

PUT ✕

**To make sure the invalidation is triggered only when you change the .html file we add the following to SUFFIX**



Suffix - *optional*

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

.html

Recursive invocation

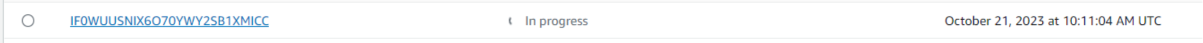
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more

☑ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

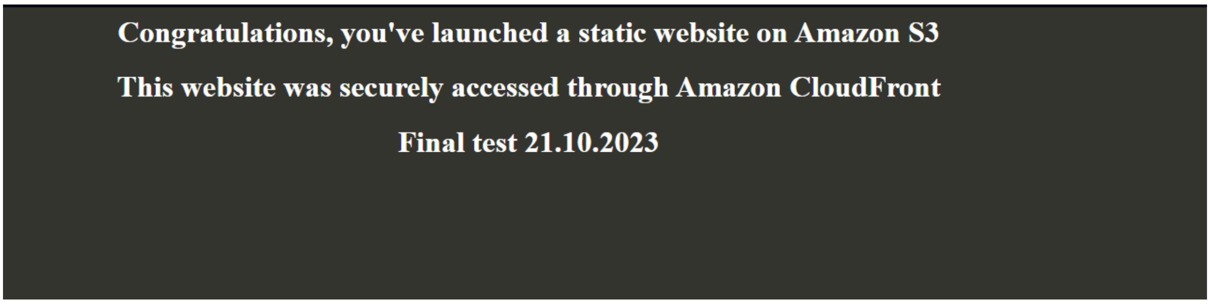
## 6.5 Final test

**I changed the code in index.html file and the lambda created the invalidation successfully:**



○ IFOWUUSNIX6O70YWY2SB1XMICC In progress October 21, 2023 at 10:11:04 AM UTC

**The website was updated within seconds:**



Congratulations, you've launched a static website on Amazon S3

This website was securely accessed through Amazon CloudFront

Final test 21.10.2023

# 7. Conclusion

In this tutorial, we've seen how you can keep your data safe. S3 buckets, which store your website files, should never be left open to the public (unless we are required to do so) because there are always people looking for exposed data. We can protect them using methods like Origin Access Control and by making sure services don't need to go over the public internet to access these files.

We've also learned about a handy way to update your website automatically. This is done using AWS CodePipeline, but there are other services like AWS CodeCommit (in our case GitHub) and AWS CodeDeploy that help you manage your code and deploy it to different parts of AWS.

Lastly, we explored an easy way to automate tasks by using AWS Lambda. This keeps your website up to date without having to manually make changes to CloudFront.

So, in simple terms, our setup emphasizes the need for safety, easy updates, and automation in your AWS projects. These practices will make your system more secure, efficient, and reliable.

**Thank you for following this tutorial. In case you need any help or guidance don’t hesitate to contact me.**
**www.paulciurean.com**