Academic Year 2019/2020

Semester I

CST 333 - Distributed & Grid Computing

Assignment 2 - Distributed Cloud Computing

No.	Name	Matric	Email
1	Muhammad Iqrar Amin	140659	iqraramin@student.usm.my
2	Tan Lee Sing	141097	tanleesing1998@student.usm.my

Due Date: 26th January

Table of Contents

1 Project Overview

We have hosted a static HTML website in an S3 bucket. The website uses CloudFront for its Content Delivery Network (CDN), which caches its contents to optimize performance. Whenever we update an object in the bucket, we will need to invalidate (purge) the object's cache so that visitors to our website will see the updated content.

1.1 Services Used

Table 1.1 shows all the services that our project is using to create the static application that will automatically invalidate the CDN cache.

Table 1.1: AWS Services Description

No.	Logo	Name	Description
1		Route 53	Route 53 is used to automatically handle the requests that go in from Internet to the internal resources of AWS in our case, CloudFront.
2		CloudFront	Amazon CloudFront is a worldwide content delivery network (CDN) service that is used to delivers data securely and with low latency.
3		S3	An S3 bucket is utilized to store all the code for our static website which is essentially a resume/portfolio website.
4		Certification Manager	By using Certificate Manager, we will create an SSL certification for our static website, so the traffic is server using HTTPS protocol.
5		Lambda	Lambda functions provide the necessary backend code that will be used to automatically invalidate the cache by sending a request to CloudFront.
6		EventBridge	EventBridge is used to create an event which will trigger the lambda function whenever a new file is added or deleted S3 bucket.

1.2 System Architecture

Figure 1.1 shows the overall architecture our project. There are two main use cases in this project. The first one is when the user of the static website wants to access it. In this case, the request from users will travel through the Route 53 first which will direct the request to the CloudFront where all the resource for our static website is cached for all unchanged requests. In case the contents are not cached, they are directly taken from S3 bucket and served to the user. Moreover, the connection from client to the CloudFront and ultimately to the S3 bucket is encrypted using SSL Certificate from AWS Certification Manager (ACM). The second use case is when a developer/programmer modifies the static website in the S3 bucket, it will trigger an event defined in the EventBridge which is essentially a Lambda function that will automatically invalidates the cache in the CloudFront CDN so that the latest content in the S3 bucket is served to the users of static website.

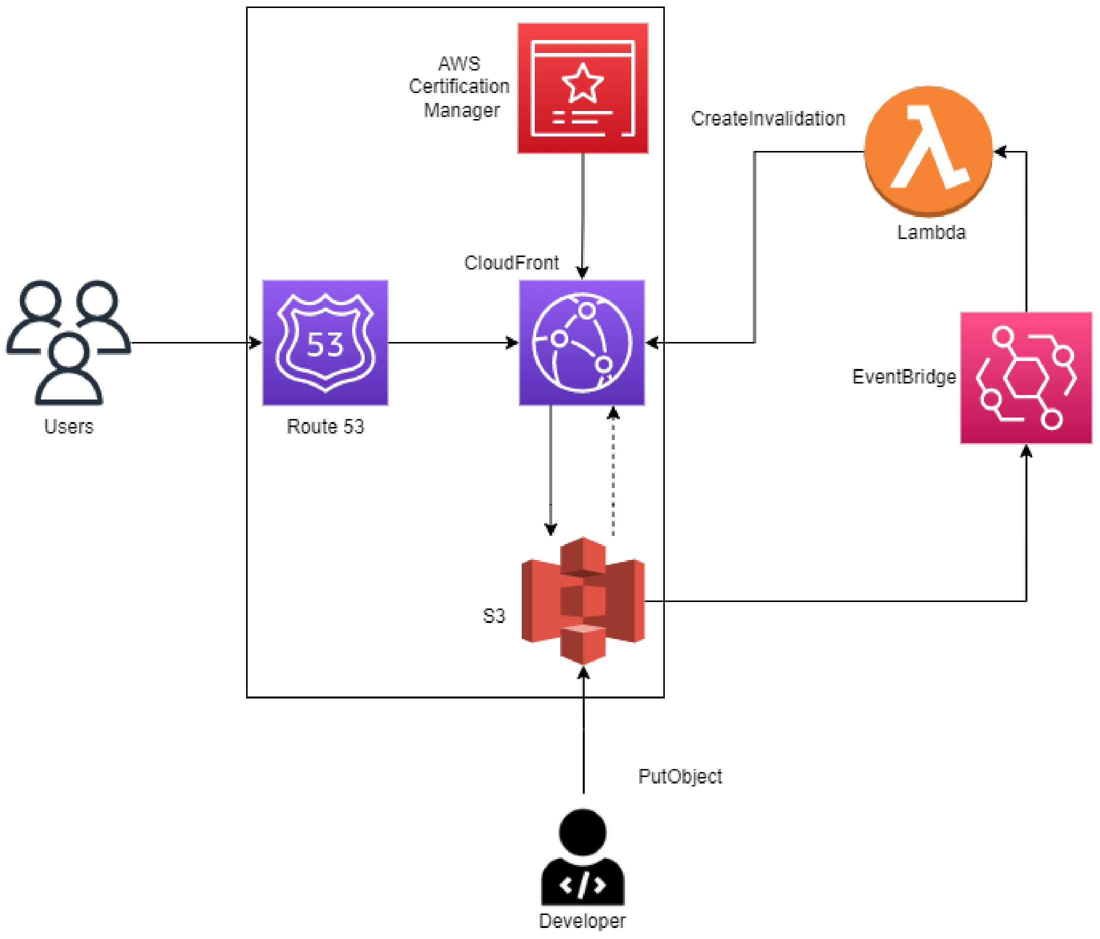
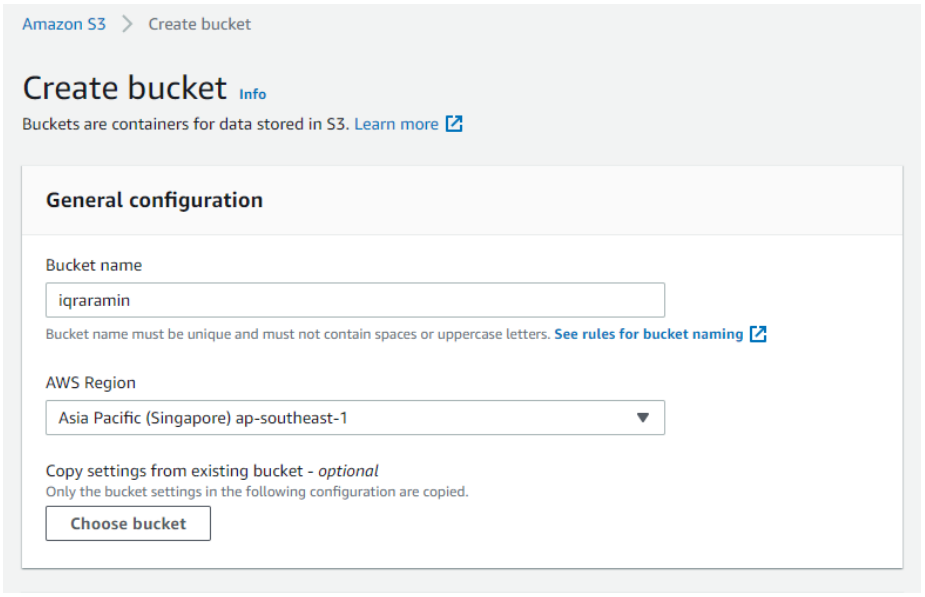


Figure 1.1: System Architecture Diagram

2 Project Environment Configuration and Settings

2.1 Amazon S3 Bucket

For our project, we first started with configuring the Amazon S3 bucket, we used the general configuration shown in the following image:



Amazon S3 > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name

iqraramin

Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region

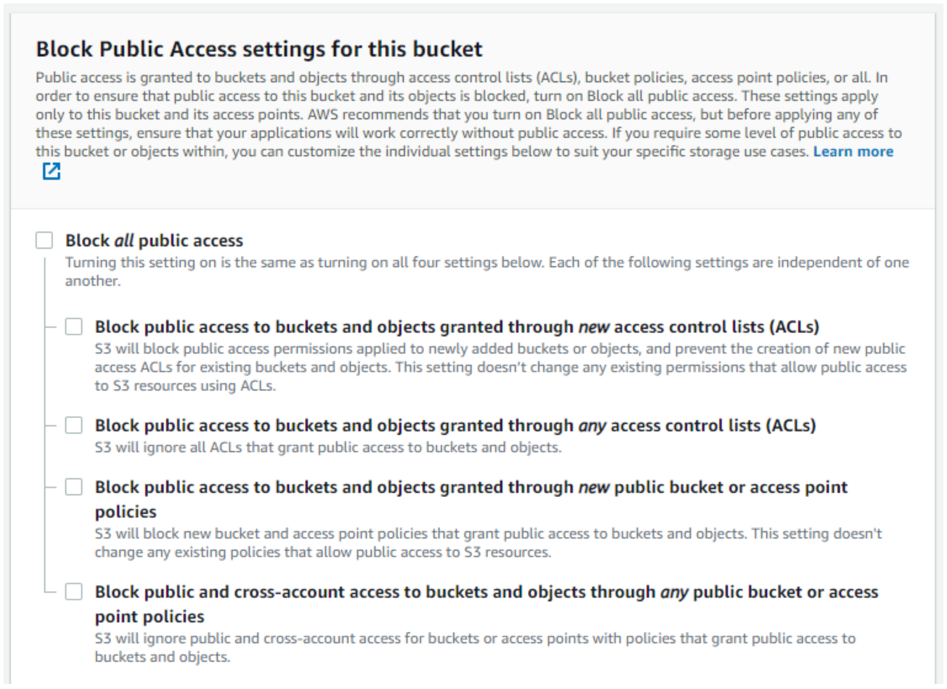
Asia Pacific (Singapore) ap-southeast-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

then we choose to allow all the public access on this bucket since this will be distributed globally to everyone on the internet, with the rest of the options being default:



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☐ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

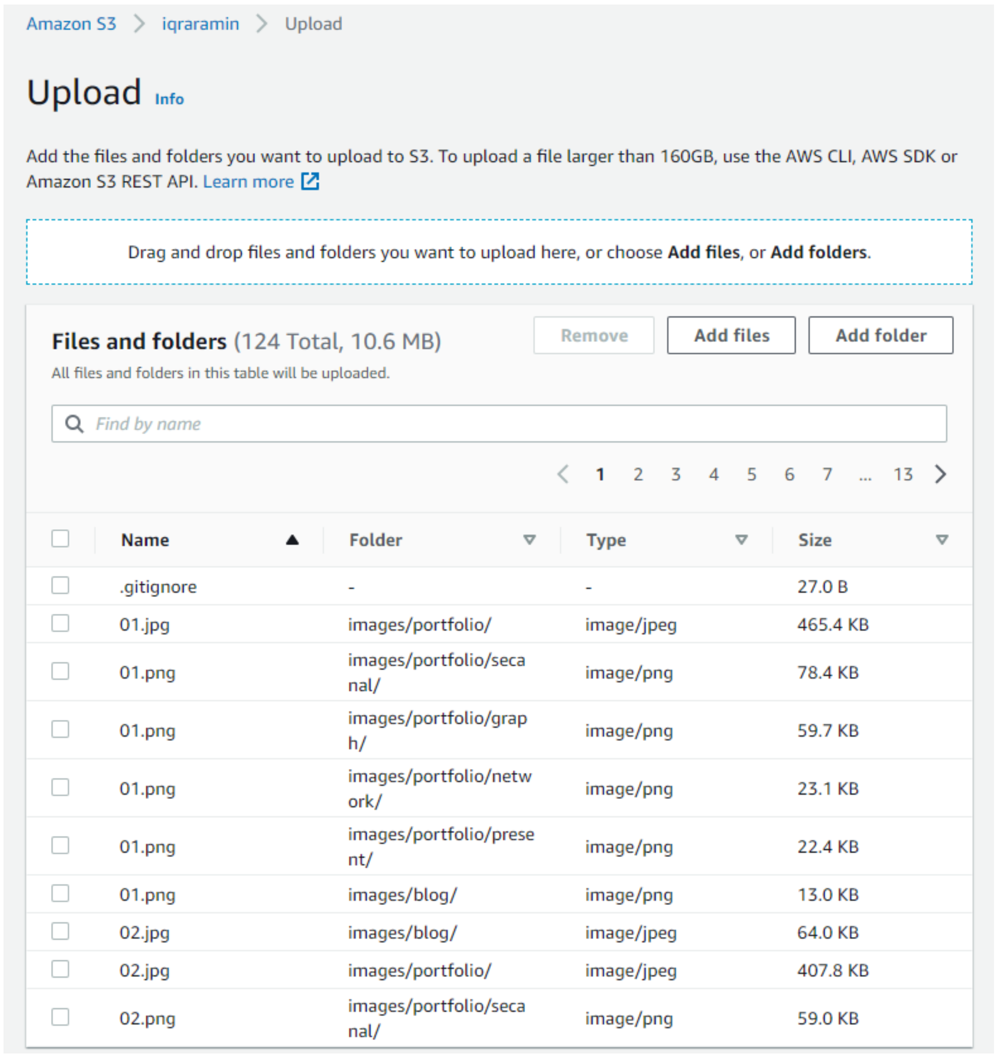
☐ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Once we created the bucket, we now upload all our code for the static website in the S3 bucket, the following is the configuration we choose to upload our content:



Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

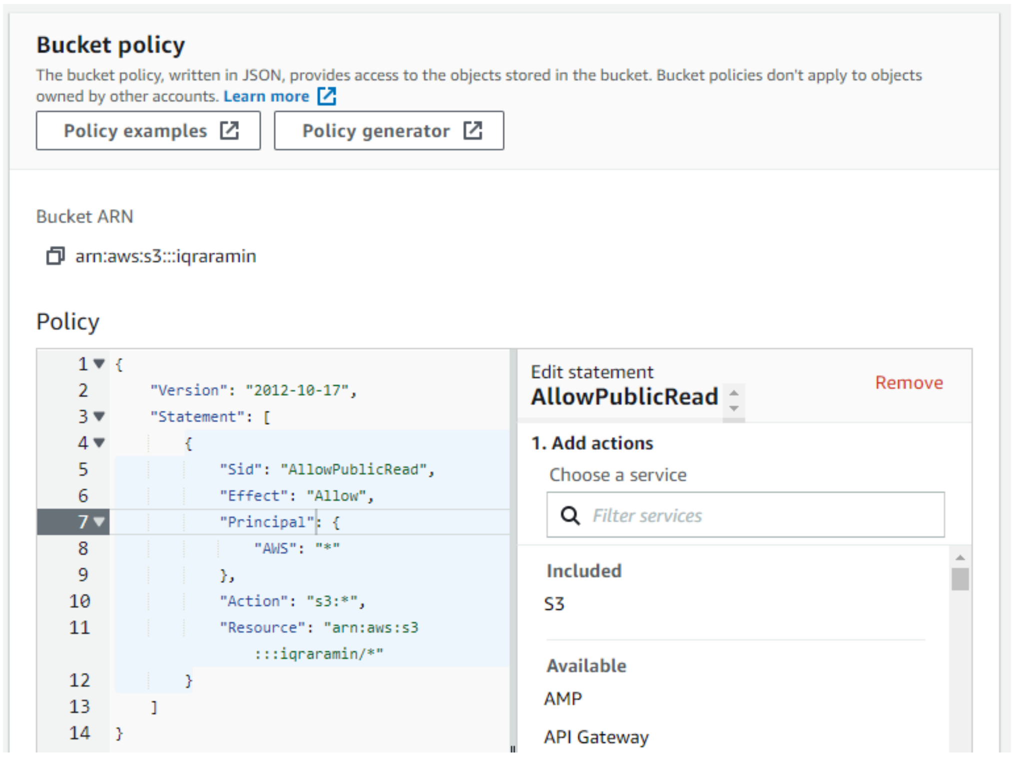
Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (124 Total, 10.6 MB)

All files and folders in this table will be uploaded.

Name	Folder	Type	Size
.gitignore	-	-	27.0 B
01.jpg	images/portfolio/	image/jpeg	465.4 KB
01.png	images/portfolio/secanal/	image/png	78.4 KB
01.png	images/portfolio/graph/	image/png	59.7 KB
01.png	images/portfolio/network/	image/png	23.1 KB
01.png	images/portfolio/present/	image/png	22.4 KB
01.png	images/blog/	image/png	13.0 KB
02.jpg	images/blog/	image/jpeg	64.0 KB
02.jpg	images/portfolio/	image/jpeg	407.8 KB
02.png	images/portfolio/secanal/	image/png	59.0 KB

Now that we have uploaded our content to the S3 bucket, we need to give appropriate permissions in the Bucket Policy, the following is the config for newly added policy:



Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN

arn:aws:s3:::iqraramin

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPublicRead",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": "arn:aws:s3
                :::iqraramin/*"
        }
    ]
}
```

Edit statement

AllowPublicRead

1. Add actions

Choose a service

Included

S3

Available

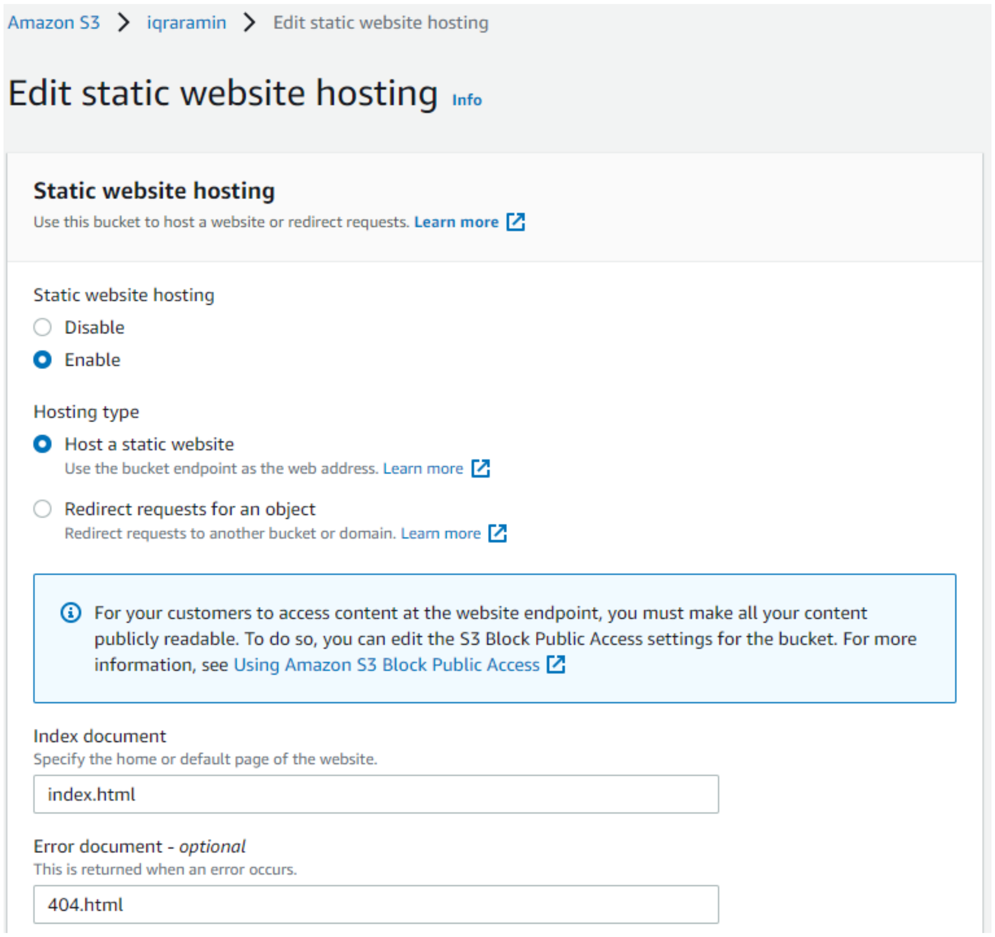
AMP

API Gateway

The policy essentially means that all the public traffic trying to access this resource will be granted access to view the content of the bucket, here is the JSON used to configure it:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::iqraramin/*"
    }
  ]
}
```

Lastly, we will enable the static website hosting in the Properties of the S3 bucket, we used *index.html* as the home page and *404.html* as error page, the following is the configuration used:



Amazon S3 > iqraramin > Edit static website hosting

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

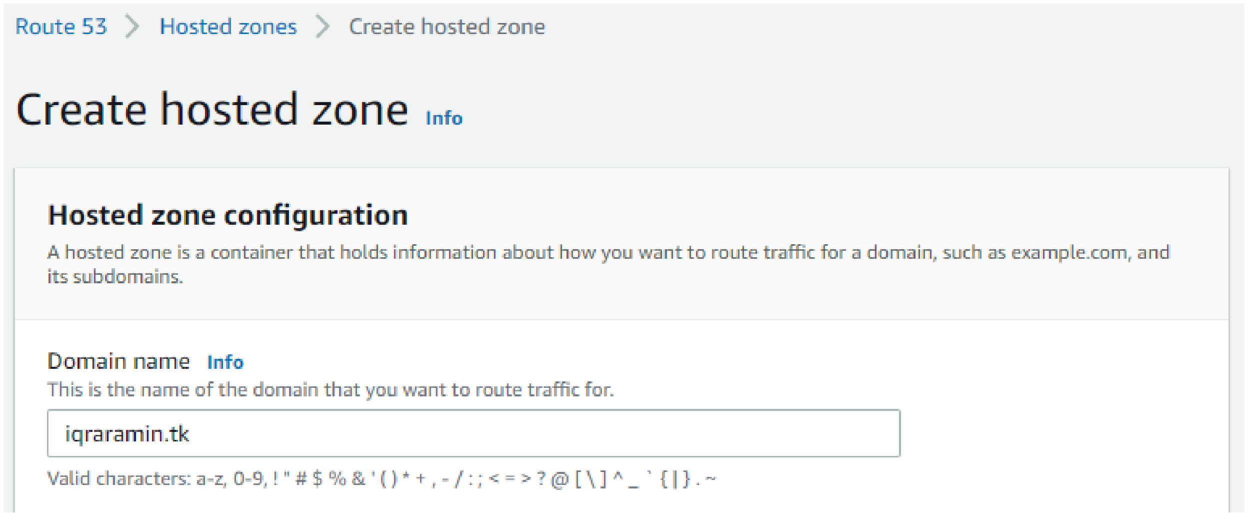
For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access

Index document
Specify the home or default page of the website.
index.html

Error document - *optional*
This is returned when an error occurs.
404.html

2.2 Amazon Route 53

Now that the initial setup for S3 bucket is complete, we will create a hosted zone for our domain name in the Route 53, the following is the hosted zone configuration:



Route 53 > Hosted zones > Create hosted zone

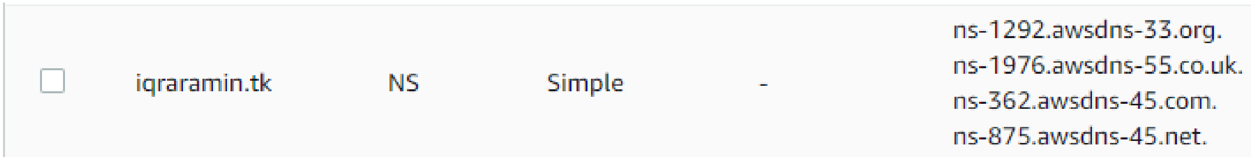
Create hosted zone Info

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

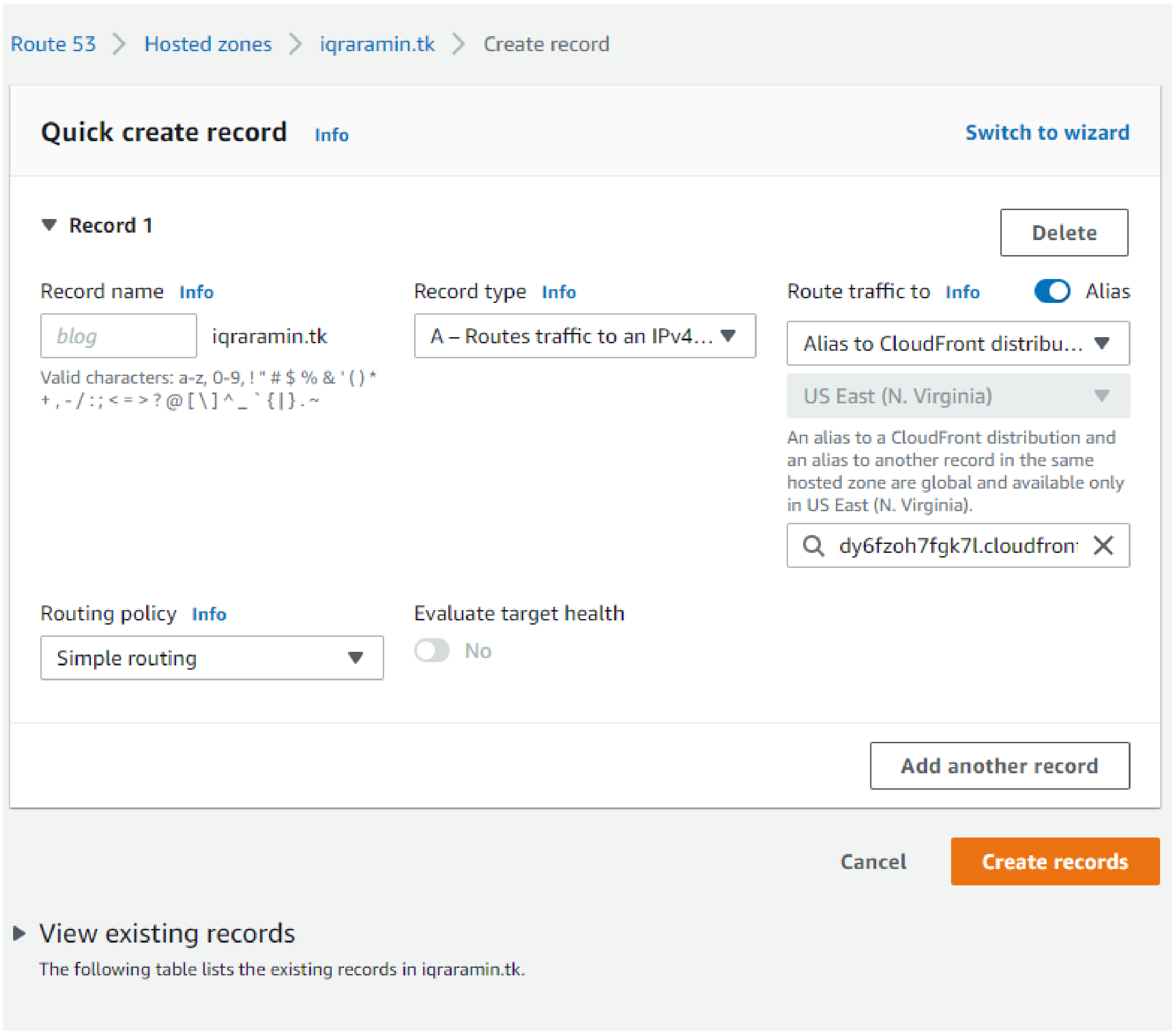
Domain name Info
This is the name of the domain that you want to route traffic for.
iqraramin.tk
Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

we will then use the nameservers created in this hosted zone in the domain name manager so that both domain name and our resources are connected, the following are the nameservers:



☐	iqraramin.tk	NS	Simple	-	ns-1292.awsdns-33.org. ns-1976.awsdns-55.co.uk. ns-362.awsdns-45.com. ns-875.awsdns-45.net.

Once we create a CloudFront distribution for our static website, we will need to add another record in hosted zone so that the traffic from the distribution will be directed to the domain, the following is the config used to create the record:



Route 53 > Hosted zones > iqraramin.tk > Create record

Quick create record Info Switch to wizard

Record 1 Delete

Record name Info: blog iqraramin.tk
Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Record type Info: A – Routes traffic to an IPv4…

Route traffic to Info: Alias
Alias to CloudFront distribu…
US East (N. Virginia)
An alias to a CloudFront distribution and an alias to another record in the same hosted zone are global and available only in US East (N. Virginia).
dy6fzoh7fgk7l.cloudfron

Routing policy Info: Simple routing

Evaluate target health: No

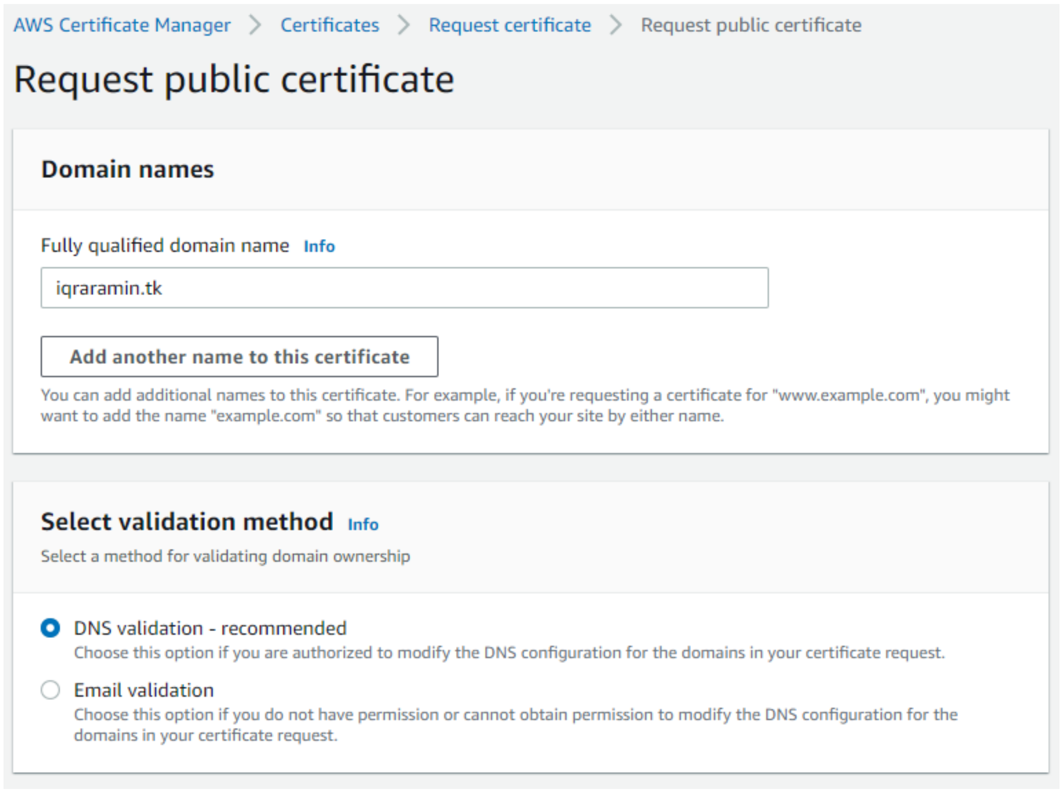
Add another record

Cancel Create records

View existing records
The following table lists the existing records in iqraramin.tk.

2.3 Amazon Certificate Manager

After setting up the appropriate routes for the static website, we will secure the all the connection to the site using SSL certificate provided by Amazon Service. We will create a public certificate as the website is publicly hosted, the following is the config used:



AWS Certificate Manager > Certificates > Request certificate > Request public certificate

Request public certificate

Domain names

Fully qualified domain name Info

iqraramin.tk

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.

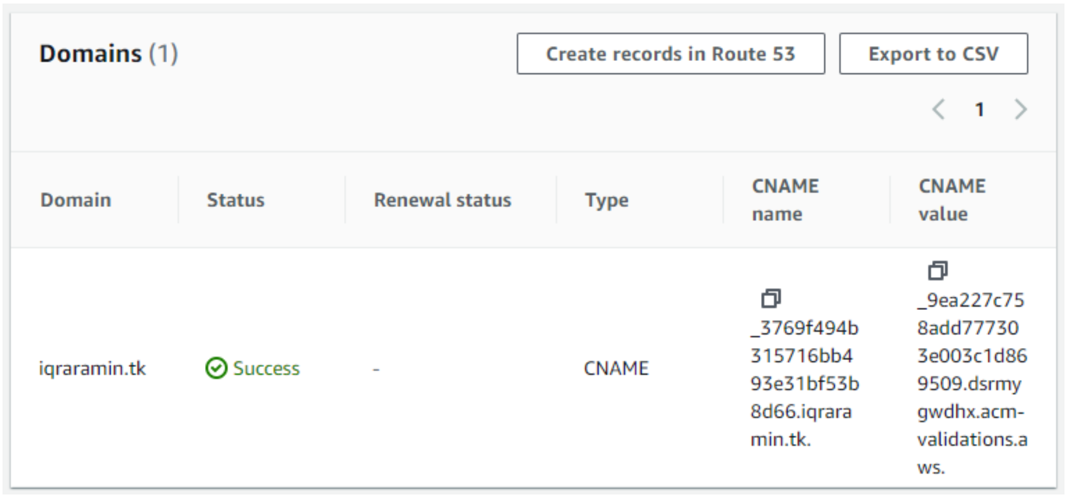
Select validation method Info

Select a method for validating domain ownership

[✓ DNS validation - recommended]
Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request.

Email validation
Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request.

Once the certificate is created, we need to create a record in the Route 53 for it to validate the content of our website, the following settings were used to create a CNAME record in Route 53:



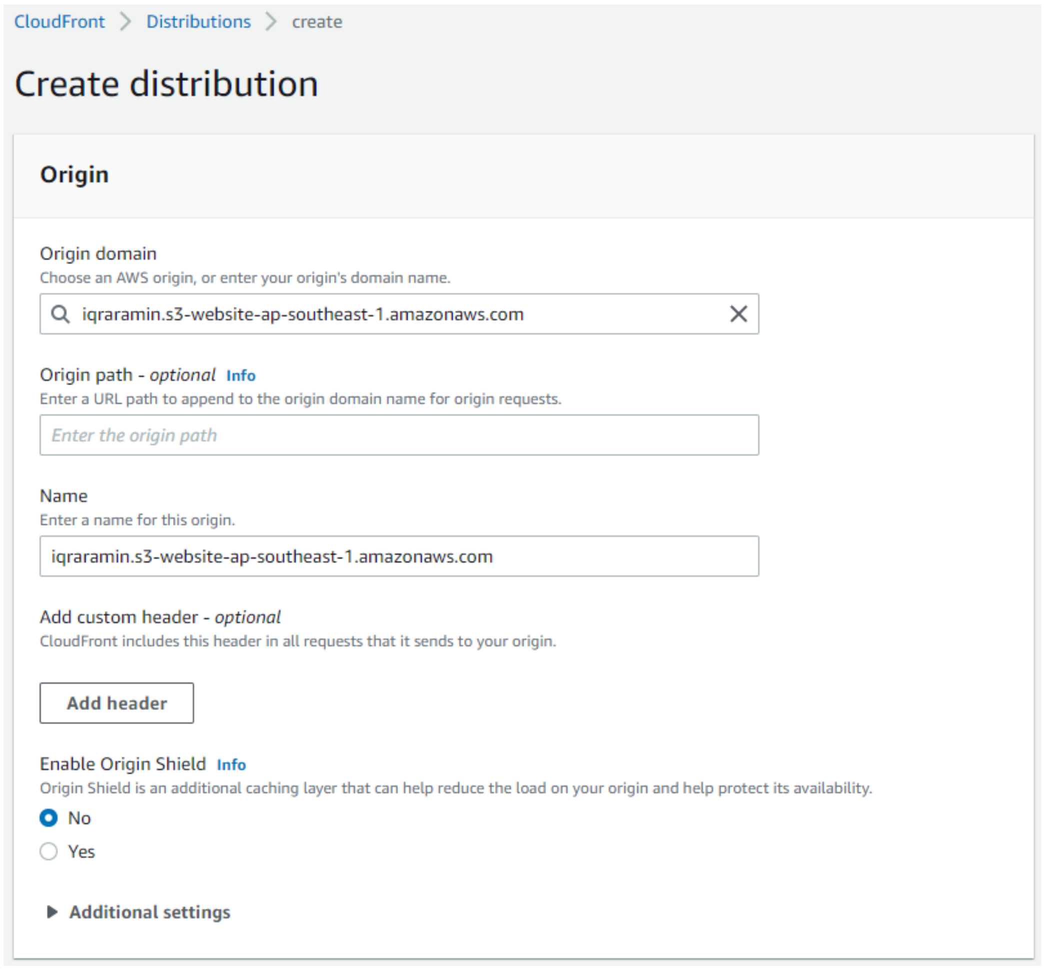
Domains (1)

Create records in Route 53 | Export to CSV

Domain	Status	Renewal status	Type	CNAME name	CNAME value
iqraramin.tk	✓ Success	-	CNAME	_3769f494b315716bb493e31bf53b8d66.iqraramin.tk.	_9ea227c758add777303e003c1d869509.dsrmygwdhx.acm-validations.aws.

2.4 Amazon CloudFront

After creating an SSL certificate using Certificate Manager, we will create a CloudFront distribution. For "Origin", we choose the domain same as the URL for our static website on S3 bucket, and the Name was auto generated by AWS. The following are the configuration for "Origin" part of the dist.:



CloudFront > Distributions > create

Create distribution

Origin

Origin domain
Choose an AWS origin, or enter your origin's domain name.
iqraramin.s3-website-ap-southeast-1.amazonaws.com

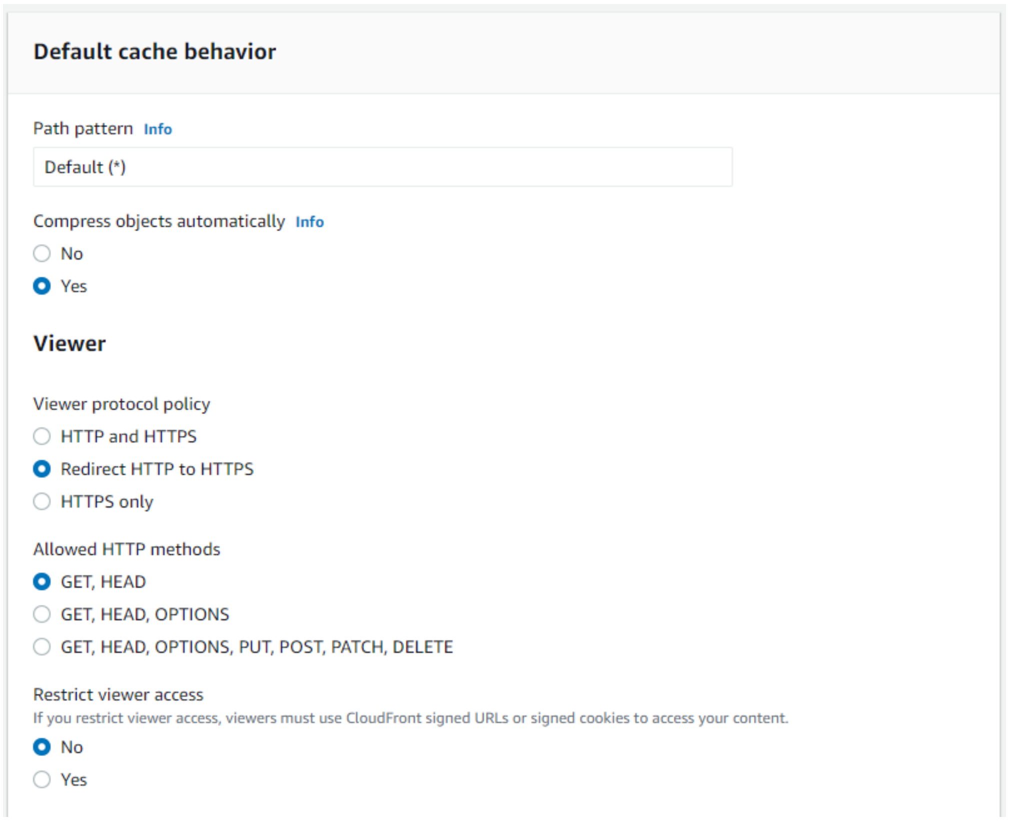
Origin path - *optional* Info
Enter a URL path to append to the origin domain name for origin requests.
Enter the origin path

Name
Enter a name for this origin.
iqraramin.s3-website-ap-southeast-1.amazonaws.com

Add custom header - *optional*
CloudFront includes this header in all requests that it sends to your origin.

Enable Origin Shield Info
Origin Shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.
[✓ No] Yes

Moving on the "Default cache behavior" part of the configuration, we only require changing on part and the remaining part will be left as the default. We only change the *Viewer protocol policy* from "HTTP and HTTPS" to "Redirect HTTP to HTTPS" since we only want a to establish secure connection, so we redirect all the unsecure connection to the secure channel. The following are the configuration for "Default cache behavior":


Default cache behavior

Path pattern Info

Default (*)

Compress objects automatically Info

No [✓ Yes]

Viewer

Viewer protocol policy

HTTP and HTTPS [✓ Redirect HTTP to HTTPS] HTTPS only

Allowed HTTP methods

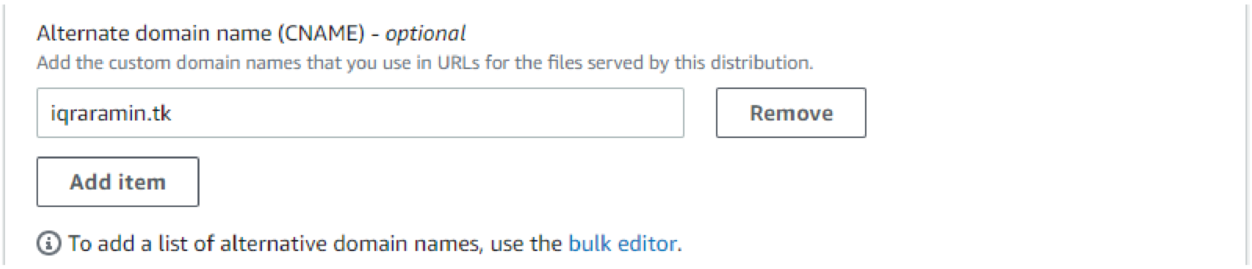
[✓ GET, HEAD] GET, HEAD, OPTIONS GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Restrict viewer access

If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.

[✓ No] Yes

In the “Settings” section of the distribution configuration, we only need to change two things first is the *Alternative domain name (CNAME)* which is used to add the custom domain names that we use in URLs for the files served by this distribution. The following is the config used:

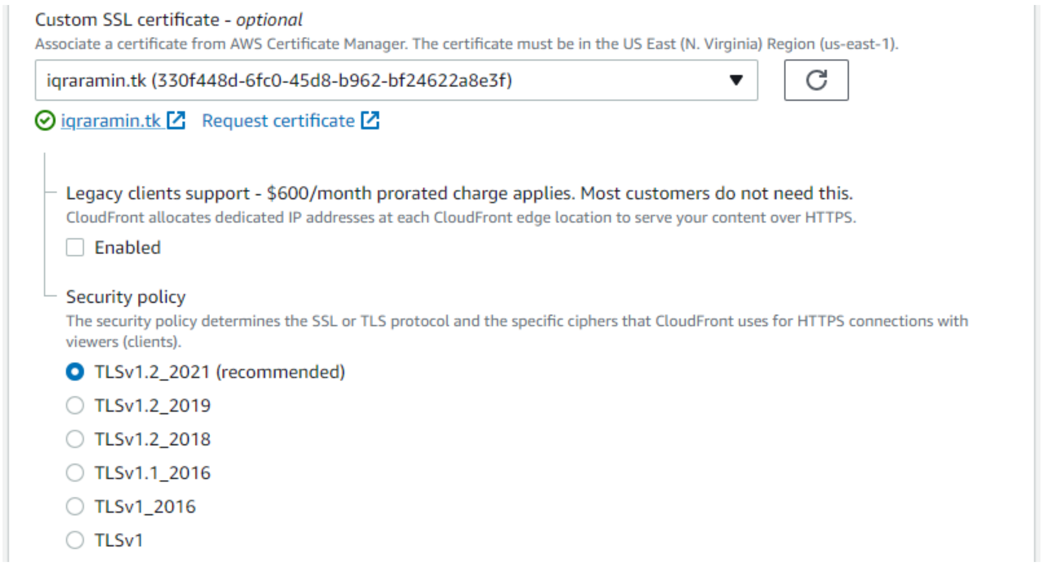

Alternate domain name (CNAME) - *optional*

Add the custom domain names that you use in URLs for the files served by this distribution.

iqraramin.tk

To add a list of alternative domain names, use the bulk editor.

Secondly, we need to add a SSL certificate that we created previously using Amazon Certificate Manager, this will associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1) hence we create the previous certificate in the same Region. The following are the configs for this:

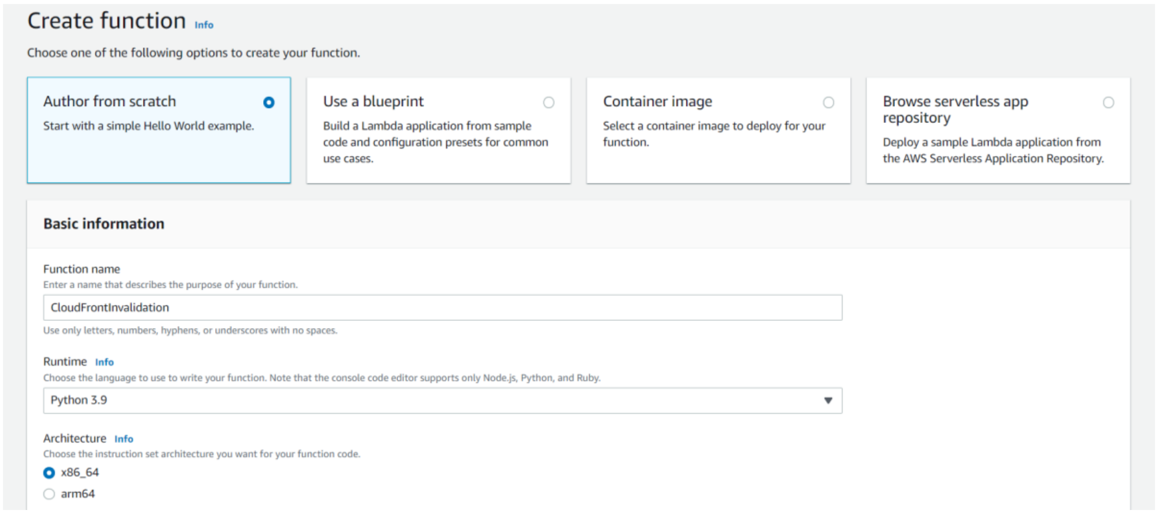

Custom SSL certificate - *optional*
Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).
iqraramin.tk (330f448d-6fc0-45d8-b962-bf24622a8e3f)
iqraramin.tk Request certificate

Legacy clients support - $600/month prorated charge applies. Most customers do not need this.
CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.
☐ Enabled

Security policy
The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).
[✓ TLSv1.2_2021 (recommended)] TLSv1.2_2019 TLSv1.2_2018 TLSv1.1_2016 TLSv1_2016 TLSv1

2.5 Lambda Function

After setting up Amazon S3 bucket and Amazon CloudFront, we will create a Lambda function for automatic invalidation whenever a new object is created or deleted. In creating function, we choose author from scratch and the programming language that we choose is Python.


Create function Info
Choose one of the following options to create your function.

[✓ 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.
Browse serverless app repository — Deploy a sample Lambda application from the AWS Serverless Application Repository.

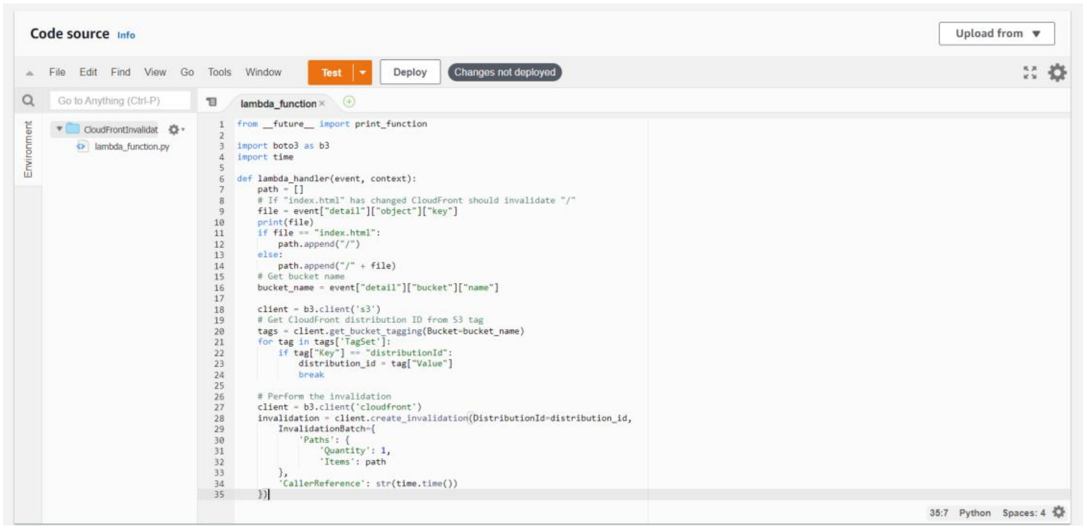
Basic information

Function name
Enter a name that describes the purpose of your function.
CloudFrontInvalidation
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

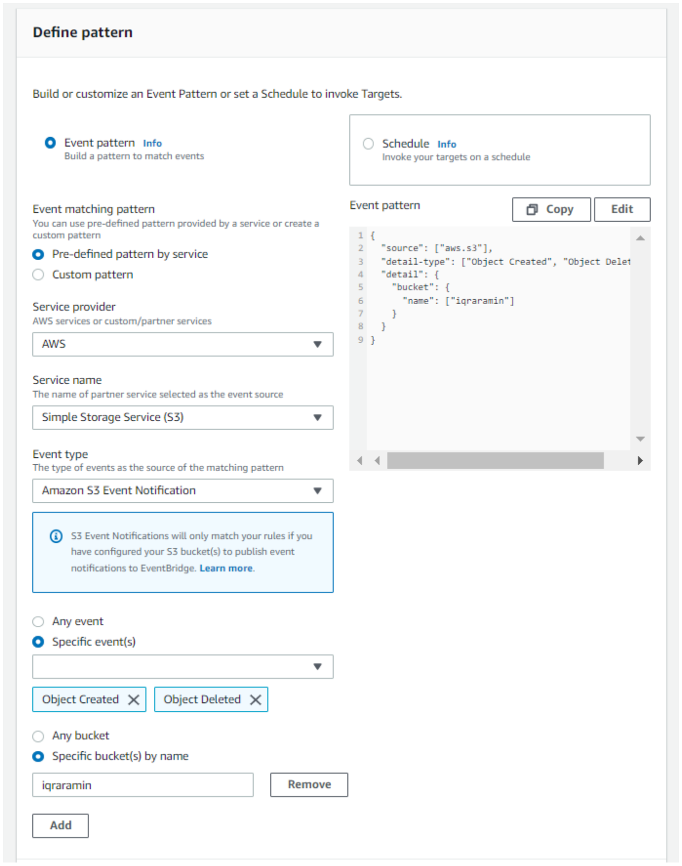
Then, in the code source, we will type in our code and deploy the code.



```python
from __future__ import print_function

import boto3 as b3
import time

def lambda_handler(event, context):
    path = []
    # If "index.html" has changed CloudFront should invalidate "/"
    file = event["detail"]["object"]["key"]
    print(file)
    if file == "index.html":
        path.append("/")
    else:
        path.append("/" + file)
    # Get bucket name
    bucket_name = event["detail"]["bucket"]["name"]

    client = b3.client('s3')
    # Get CloudFront distribution ID from S3 tag
    tags = client.get_bucket_tagging(Bucket=bucket_name)
    for tag in tags['TagSet']:
        if tag["Key"] == "distributionId":
            distribution_id = tag["Value"]
            break

    # Perform the invalidation
    client = b3.client('cloudfront')
    invalidation = client.create_invalidation(DistributionId=distribution_id,
        InvalidationBatch={
            'Paths': {
                'Quantity': 1,
                'Items': path
        },
        'CallerReference': str(time.time())
    })
```

2.6 Amazon EventBridge

After the function is created, we create a new rule in Amazon EventBridge to trigger the function in Lambda. For define pattern, we choose event pattern and not schedule as we want to trigger function in Lambda every time a new object is created or deleted. Then, we will choose the pre-defined pattern by service and below is the configuration for the event pattern.



Build or customize an Event Pattern or set a Schedule to invoke Targets.

[✓ Event pattern] Build a pattern to match events
Schedule — Invoke your targets on a schedule

Event matching pattern
You can use pre-defined pattern provided by a service or create a custom pattern
[✓ Pre-defined pattern by service]
Custom pattern

Service provider
AWS services or custom/partner services
AWS

Service name
The name of partner service selected as the event source
Simple Storage Service (S3)

Event type
The type of events as the source of the matching pattern
Amazon S3 Event Notification

S3 Event Notifications will only match your rules if you have configured your S3 bucket(s) to publish event notifications to EventBridge. Learn more.

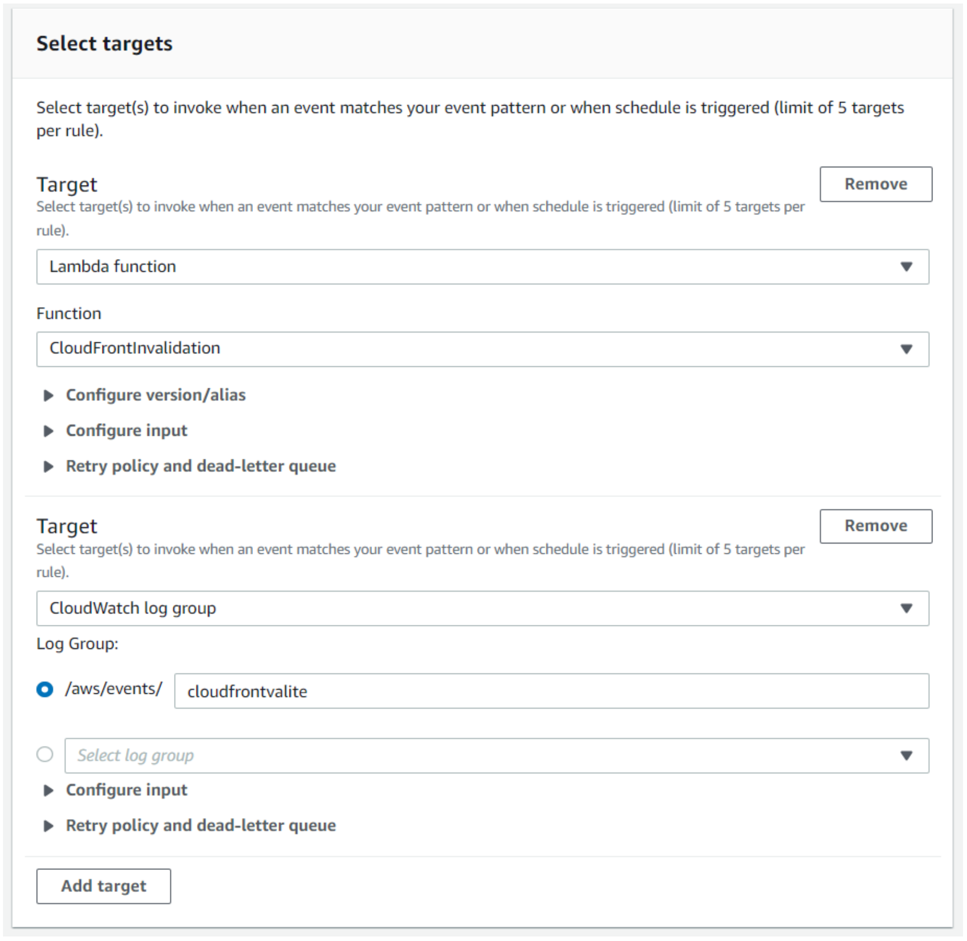
Any event
[✓ Specific event(s)]
Object Created, Object Deleted

Any bucket
[✓ Specific bucket(s) by name]
iqraramin

Event pattern

```json
{
  "source": ["aws.s3"],
  "detail-type": ["Object Created", "Object Delet
  "detail": {
    "bucket": {
      "name": ["iqraramin"]
    }
  }
}
```

Moving on to select target, the first target, we choose Lambda function and the second target, we choose CloudWatch log group, so that we can track the log. Below is the configuration for the select targets. After that, we can create the rule.



Select targets

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

Target

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

Lambda function

Function: CloudFrontInvalidation

Target

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

CloudWatch log group

Log Group:

[✓ /aws/events/] cloudfrontvalite

Select log group

3 Lessons Learned

The following are the lessons we learned from creating a static website using Amazon Web Services, we learned how to:

1. Host static website on Amazon Bucket.
2. Route the traffic of CloudFront CDN to the domain name using Route 53.
3. Create SSL certificate using Amazon Certificate Manager.
4. Configure SSL certificate in CloudFront to use HTTPS.
5. Configure different services in Amazon Web Service
6. Configure a domain name with the CloudFront resources.
7. Use different methods to trigger Lambda function from S3.
8. Create manual and automatic invalidation in CloudFront.
9. Configure Amazon EventBridge to receive notification from S3 and target to the destination that we want.

4 Task Distribution

The following are the task distribution for this project between Muhammad Iqrar Amin and Tan Lee Sing:

Muhammad Iqrar Amin

- Research possible ways to host static website on the Amazon Web Services
- Create and configure S3 Bucket to host static website
- Create and configure Route 53 to route traffic from CDN to domain
- Create and configure SSL Certificate using Certificate Manager
- Create and configure CloudFront to act as a CDN
- Write Report section 1, 2 (2.1, 2.2, 2.3, 2.4), 3, 4
- Prepare Slides

Tan Lee Sing

- Research possible ways to automate cache invalidation in CloudFront from S3 Bucket
- Create and configure Lambda function to invalidate cache in CloudFront
- Create and configure event to trigger previously create Lambda function
- Write Report section 2 (2.5, 2.6), 3, 5, 6

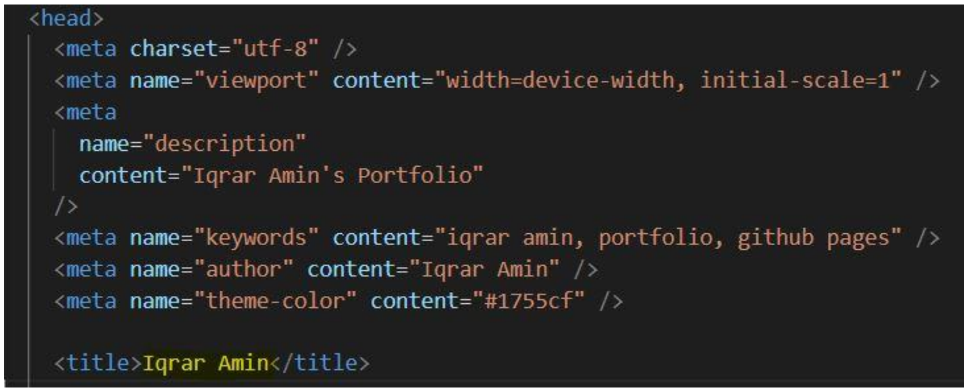
5 Source Code

```
from __future__ import print_function

import boto3 as b3
import time

def lambda_handler(event, context):
    path = []
    #CloudFront will invalidate "/" if "index.html" has changed
    file = event["detail"]["object"]["key"]
    if file == "index.html":
        path.append("/")
    else:
        path.append("/" + file)
    # Get bucket name for S3
    bucket_name = event["detail"]["bucket"]["name"]

    client = b3.client('s3')
    # Get CloudFront distribution ID from S3 tag
    tags = client.get_bucket_tagging(Bucket=bucket_name)
    for tag in tags['TagSet']:
        if tag["Key"] == "distributionId":
            distribution_id = tag["Value"]
            break

    # Perform invalidation
    client = b3.client('cloudfront')
    invalidation = client.create_invalidation(DistributionId=distribution_id,
        InvalidationBatch={
            'Paths': {
                'Quantity': 1,
                'Items': path
        },
        'CallerReference': str(time.time())
    })
```

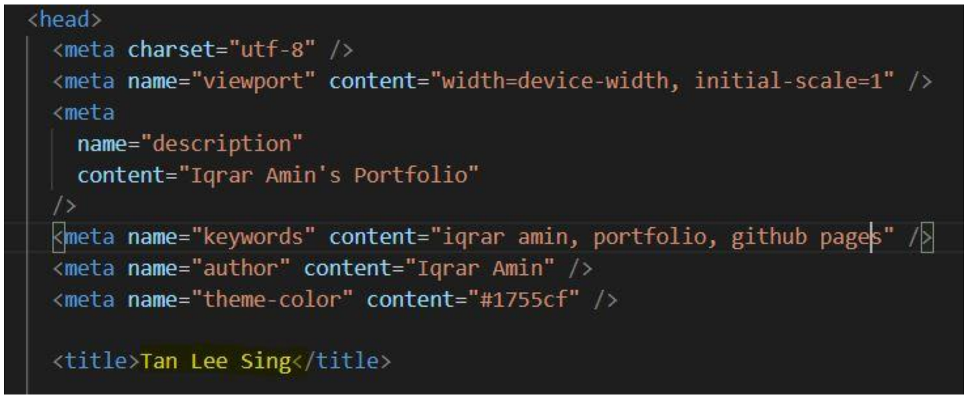
The code above is the code to invalidate CloudFront whenever a new object is created or deleted. At line 9, the code is to get the file name of the object. Then, we will check if the file name is “index.html” or another file. If the file is “index.html”, we will append “/” to the path.

At line 15, we will get the bucket name for the use to get distribution ID for our distribution in CloudFront. Then, we will invalidate CloudFront using the distribution ID that we get [1] .

6 Proof of Concept (Screenshots)

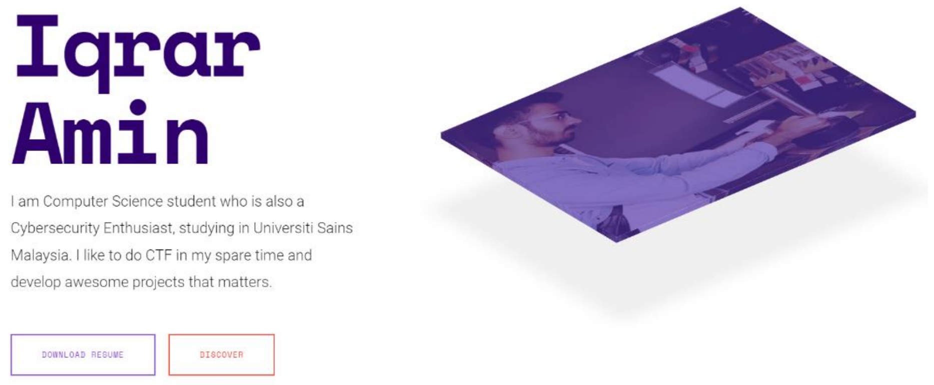


```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta
    name="description"
    content="Iqrar Amin's Portfolio"
  />
  <meta name="keywords" content="iqrar amin, portfolio, github pages" />
  <meta name="author" content="Iqrar Amin" />
  <meta name="theme-color" content="#1755cf" />

  <title>Iqrar Amin</title>
```



```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta
    name="description"
    content="Iqrar Amin's Portfolio"
  />
  <meta name="keywords" content="iqrar amin, portfolio, github pages" />
  <meta name="author" content="Iqrar Amin" />
  <meta name="theme-color" content="#1755cf" />

  <title>Tan Lee Sing</title>
```

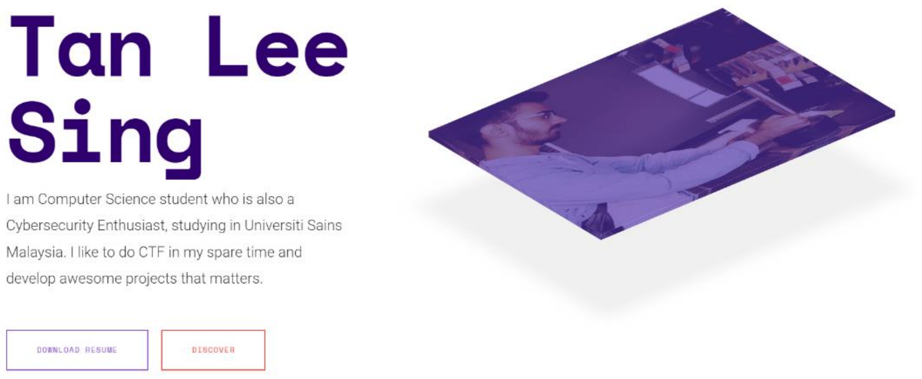
The pictures above are examples of changes that we make before and after.



Iqrar Amin

I am Computer Science student who is also a Cybersecurity Enthusiast, studying in Universiti Sains Malaysia. I like to do CTF in my spare time and develop awesome projects that matters.

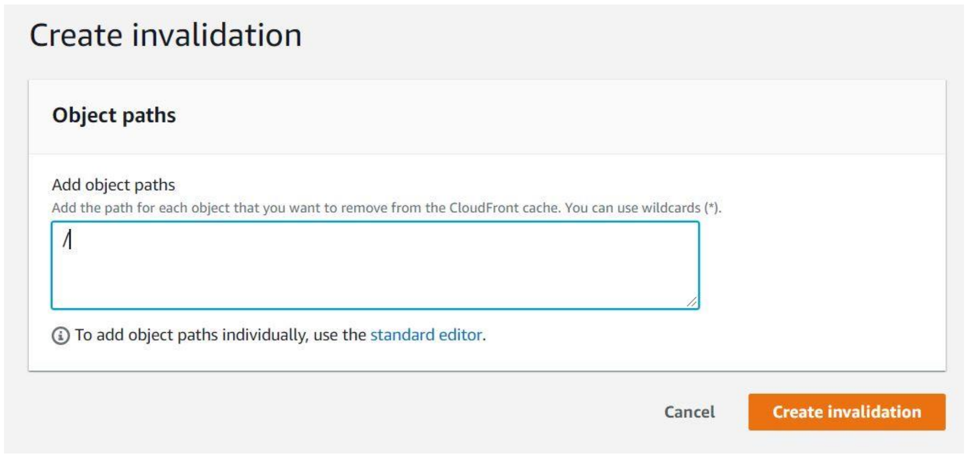
Figure above show the webpage before change.


Tan Lee Sing

I am Computer Science student who is also a Cybersecurity Enthusiast, studying in Universiti Sains Malaysia. I like to do CTF in my spare time and develop awesome projects that matters.

Figure above show the result that we expect.

Before automatic invalidation, when we do changes like update the “index.html”, the website will not update, and we need to wait 24 hours for the website to update. If we want to update the website, we need to do manual invalidation in CloudFront. Below, we show the step to do manual invalidation in CloudFront.

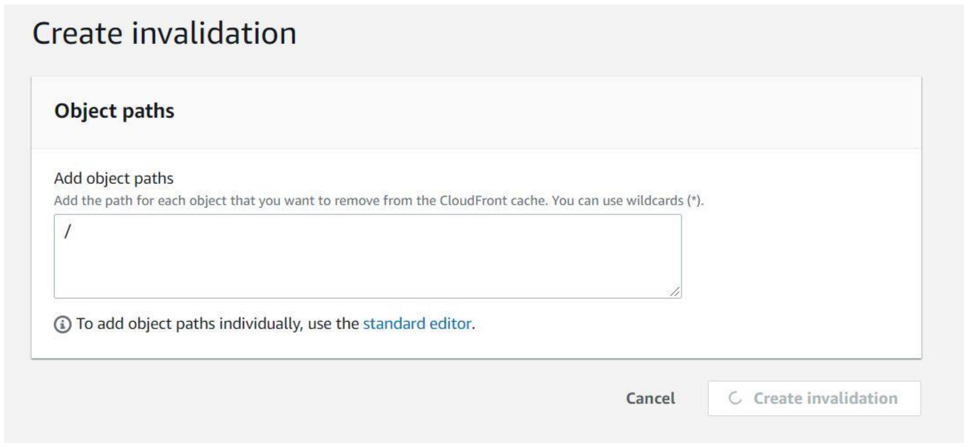

Create invalidation

Object paths

Add object paths
Add the path for each object that you want to remove from the CloudFront cache. You can use wildcards (*).

```
/
```

To add object paths individually, use the standard editor.

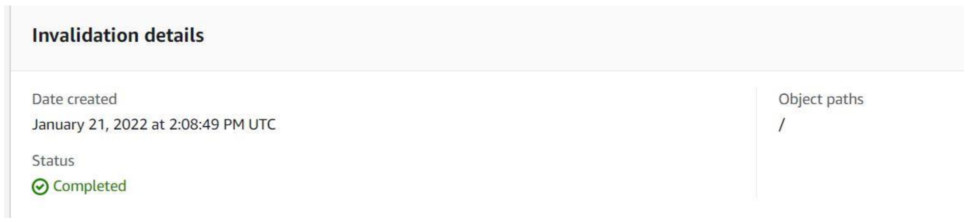

Create invalidation

Object paths

Add object paths
Add the path for each object that you want to remove from the CloudFront cache. You can use wildcards (*).

```
/
```

To add object paths individually, use the standard editor.


Invalidation details

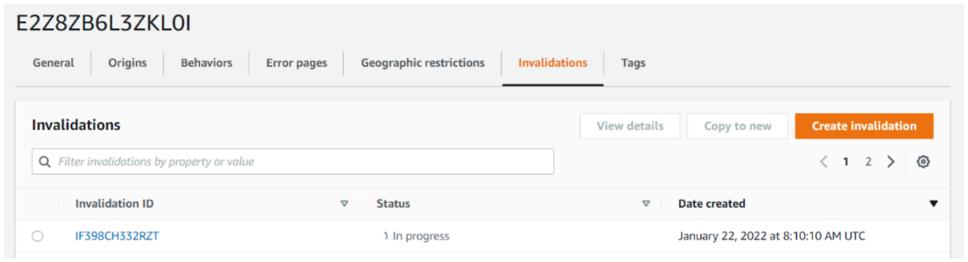
Date created
January 21, 2022 at 2:08:49 PM UTC

Status
Completed

Object paths
/

We can see from the step above; we need to put the object path where we make change and create the invalidation.

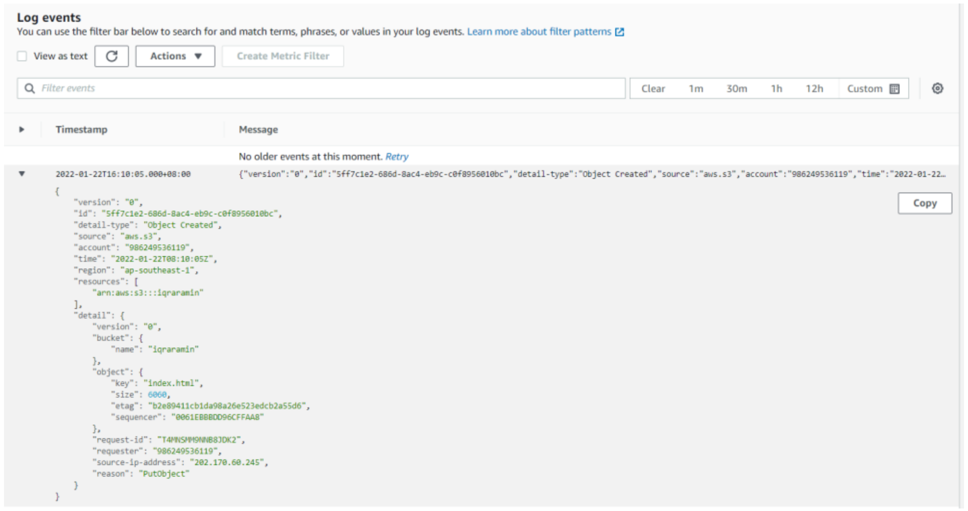
After we deploy the Lambda function for automatic invalidation, the Invalidation process will happen automatically.


E2Z8ZB6L3ZKL0I

Invalidations

Invalidation ID	Status	Date created
IF398CH332RZT	In progress	January 22, 2022 at 8:10:10 AM UTC

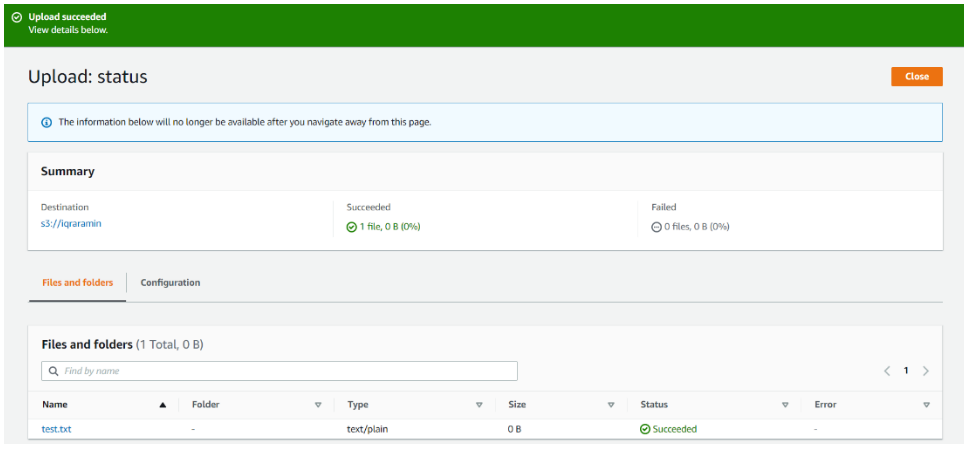
It will show that the invalidation is in progress in the invalidation tab in CloudFront.


Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2022-01-22T16:10:05.000+08:00    {"version":"0","id":"5ff7c1e2-686d-8ac4-eb9c-c0f8956010bc","detail-type":"Object Created","source":"aws.s3","account":"986249536119","time":"2022-01-22…
{
    "version": "0",
    "id": "5ff7c1e2-686d-8ac4-eb9c-c0f8956010bc",
    "detail-type": "Object Created",
    "source": "aws.s3",
    "account": "986249536119",
    "time": "2022-01-22T08:10:05Z",
    "region": "ap-southeast-1",
    "resources": [
        "arn:aws:s3:::iqraramin"
    ],
    "detail": {
        "version": "0",
        "bucket": {
            "name": "iqraramin"
        },
        "object": {
            "key": "index.html",
            "size": 6060,
            "etag": "b2e89411cb1da98a26e523edcb2a55d6",
            "sequencer": "0061EBBBDD96CFFAA8"
        },
        "request-id": "T4MNSMM9NNB8JDK2",
        "requester": "986249536119",
        "source-ip-address": "202.170.60.245",
        "reason": "PutObject"
    }
}
```

Figure above show the log event when the object is created or deleted in CloudWatch.

Next, we will show the example when we upload another file. The file that we will upload is a text document named "test.txt". Below is the example of the result that show in AWS.


Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

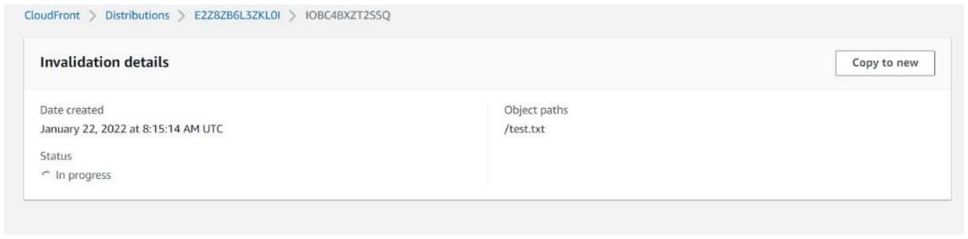
Summary

Destination	Succeeded	Failed
s3://iqraramin	1 file, 0 B (0%)	0 files, 0 B (0%)

Files and folders (1 Total, 0 B)

Name	Folder	Type	Size	Status	Error
test.txt	-	text/plain	0 B	Succeeded	-

The pictures above show the result when we upload “test.txt” into Amazon S3.

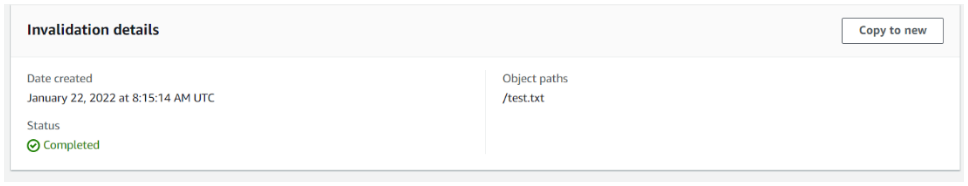

CloudFront > Distributions > E2Z8ZB6L3ZKL0I > IOBC4BXZT2SSQ

Invalidation details

Date created
January 22, 2022 at 8:15:14 AM UTC

Status
In progress

Object paths
/test.txt


Invalidation details

Date created
January 22, 2022 at 8:15:14 AM UTC

Status
Completed

Object paths
/test.txt

The pictures above show that the cache invalidation automatically got triggered and then changed the cache in the CloudFront.

References

[1] LewisCraik. [Online]. Available: https://github.com/LewisCraik/invalidate-cloudfront-on-s3-update.