

DEPLOYING WEB APPLICATION & MODEL ON AWS CLOUD USING VARIOUS SERVICES

By:

Arisha Akhtar - A041

Tina Narsinghani - A040

Introduction : Traditional on-premises web architectures require complex solutions and accurate reserved capacity forecast in order to ensure reliability. Dense peak traffic periods and wild swings in traffic patterns result in low utilization rates of expensive hardware. This yields high operating costs to maintain idle hardware, and an inefficient use of capital for underused hardware. Amazon Web Services (AWS) provides a reliable, scalable, secure, and highly performing infrastructure for the most demanding web applications. This infrastructure matches IT costs with customer traffic patterns in near-real time. AWS provides seamless and cost effective solutions.

Services used for this Case study:

1. Web application was deployed on AWS Amplify , Amazon S3 .
2. ML Model XGboost was deployed using Sagemaker, stored data in S3 , ran the code using Lambda Function for prediction using the model.

Case - 1 : Deploying webpage on cloud using AWS Amplify

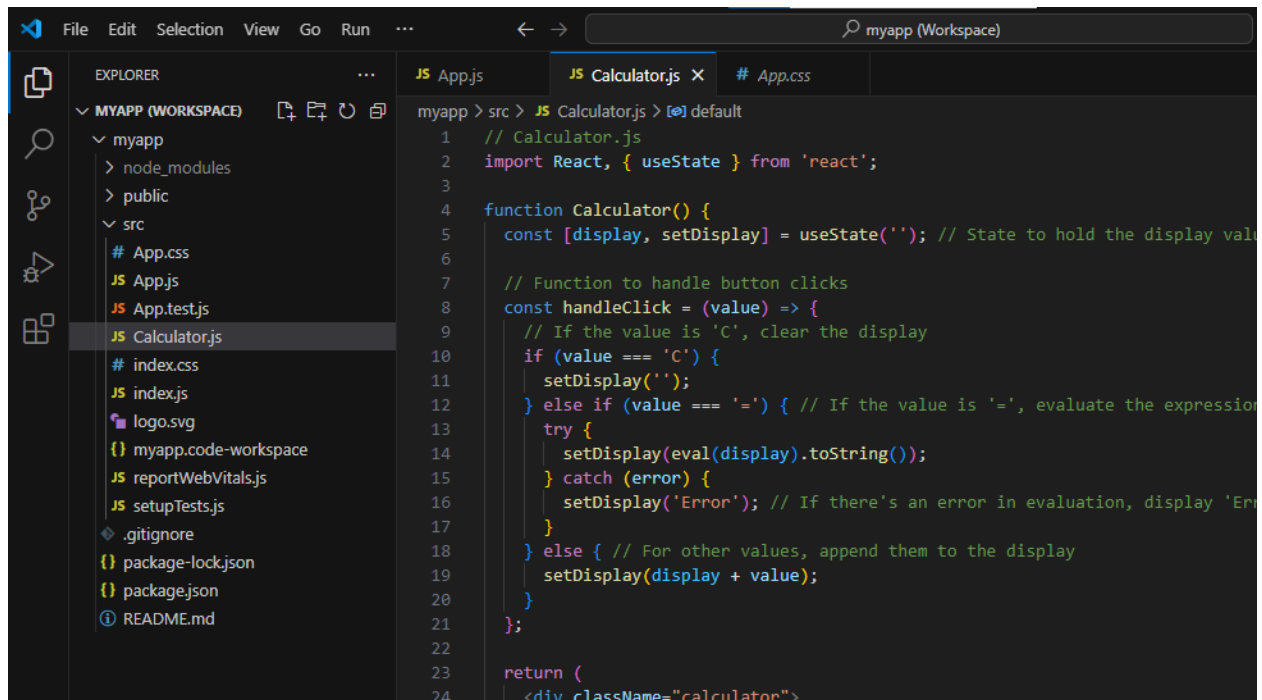
Amplify: Amplify is a framework with a set of tools and services that help you build scalable full stack applications, powered by AWS services. AWS Amplify is a set of tools and services that can be used together or on their own, to help front-end web and mobile developers build scalable full stack applications, powered by AWS. With Amplify, you can configure app backends and connect your app in minutes, deploy static web apps in a few clicks, and easily manage app content outside the AWS console.

For an example we deployed a React js web application on AWS Amplify

Methodology / Process of deployment:

Created a web application using react js on visual studio code . For deploying the webapp on Amplify we first connected to Github repository & we used Git app . Visual studio and git commands were used to push files to the github repository.

Step 1: We had created react js files on vscode:



The screenshot shows the Visual Studio Code interface with a workspace named 'myapp'. The Explorer sidebar on the left displays the project structure: 'MYAPP (WORKSPACE)' containing 'node_modules', 'public', and 'src'. The 'src' folder contains files like 'App.css', 'App.js', 'App.test.js', 'Calculator.js', 'index.css', 'index.js', 'logo.svg', 'myapp.code-workspace', 'reportWebVitals.js', 'setupTests.js', '.gitignore', 'package-lock.json', 'package.json', and 'README.md'. The main editor area shows the 'Calculator.js' file with the following code:

```
1 // Calculator.js
2 import React, { useState } from 'react';
3
4 function Calculator() {
5   const [display, setDisplay] = useState(''); // State to hold the display value
6
7   // Function to handle button clicks
8   const handleClick = (value) => {
9     // If the value is 'C', clear the display
10    if (value === 'C') {
11      setDisplay('');
12    } else if (value === '=') { // If the value is '=', evaluate the expression
13      try {
14        setDisplay(eval(display).toString());
15      } catch (error) {
16        setDisplay('Error'); // If there's an error in evaluation, display 'Error'
17      }
18    } else { // For other values, append them to the display
19      setDisplay(display + value);
20    }
21  };
22
23  return (
24    <div className="calculator">
```

Step 2: Downloaded git and wrote these commands in Visual studio code to push the files on github repository.

```
PS C:\Users\91916\Desktop\react\myapp> git init
Initialized empty Git repository in C:/Users/91916/Desktop/react/myapp/.git/
PS C:\Users\91916\Desktop\react\myapp> git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
```

```
PS C:\Users\91916\Desktop\react\myapp> git commit -m "first commit"
[master (root-commit) 894d783] first commit
 20 files changed, 18802 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
```

```

PS C:\Users\91916\Desktop\react\myapp> git branch -M main
PS C:\Users\91916\Desktop\react\myapp> git remote add origin https://github.com/TinaHN123/Cloud-Computing-Theory-pro.git
PS C:\Users\91916\Desktop\react\myapp> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 24, done.

```

Here are the Git commands that were used to connect to our Github repository:

`git init`

`git add .`

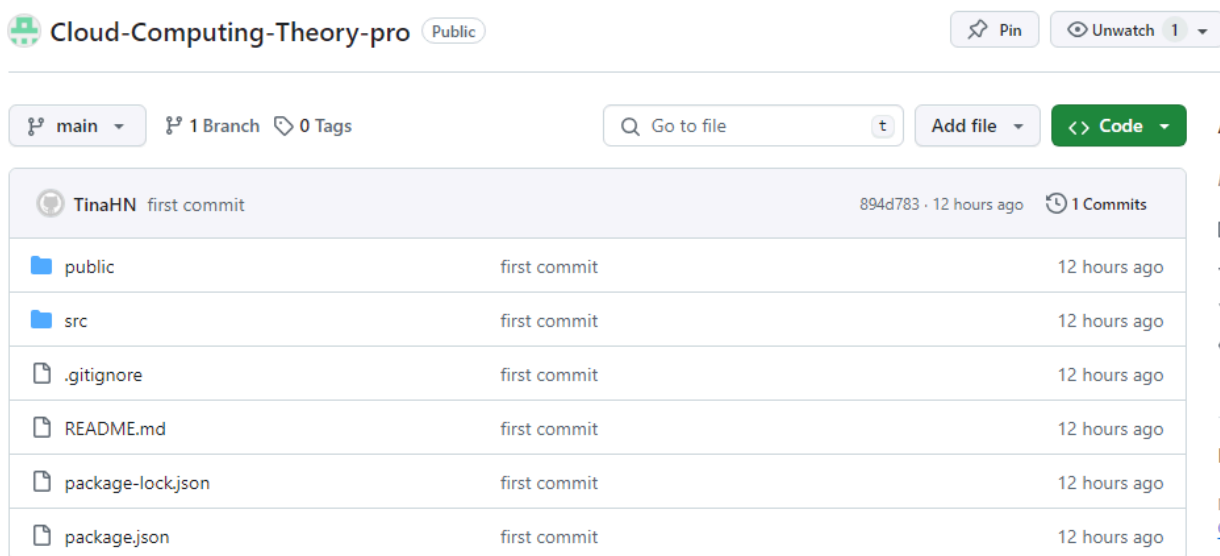
`git commit -m "first commit"`

`git branch -M main`

`git remote add origin https://github.com/TinaHN123/Practice.git`

`git push -u origin main`

Step 3: Files were successfully pushed into Github repository



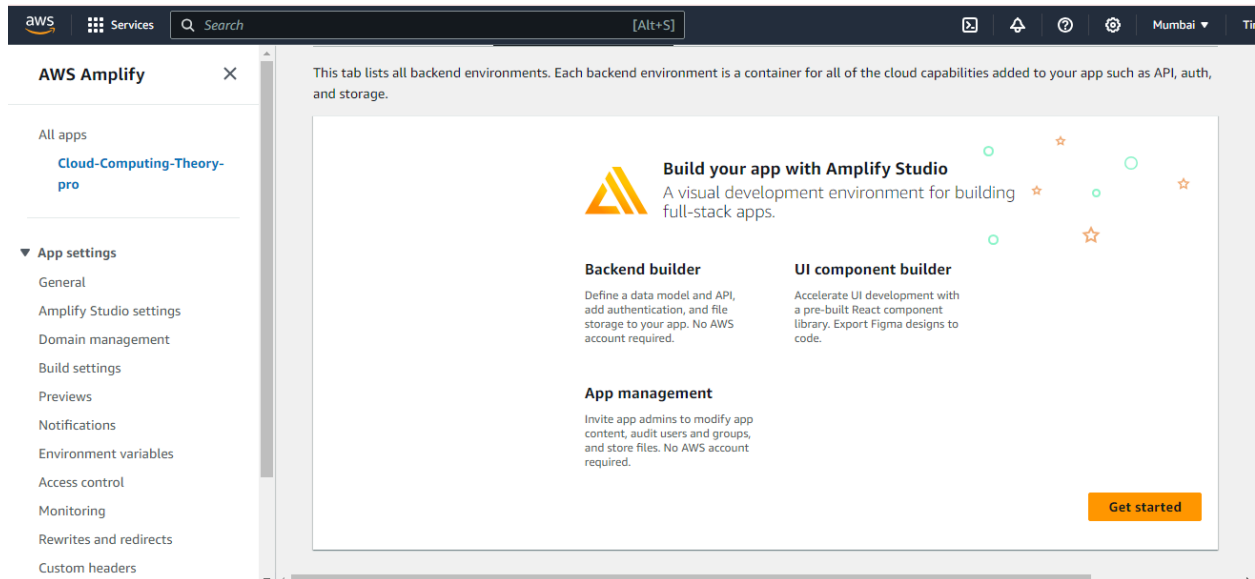
The screenshot shows the GitHub interface for a repository named "Cloud-Computing-Theory-pro" by user TinaHN. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a single commit titled "first commit" by TinaHN, made 12 hours ago. The commit details show a list of files: public, src, .gitignore, README.md, package-lock.json, and package.json, all added in the first commit.

File	Commit	Time
public	first commit	12 hours ago
src	first commit	12 hours ago
.gitignore	first commit	12 hours ago
README.md	first commit	12 hours ago
package-lock.json	first commit	12 hours ago
package.json	first commit	12 hours ago

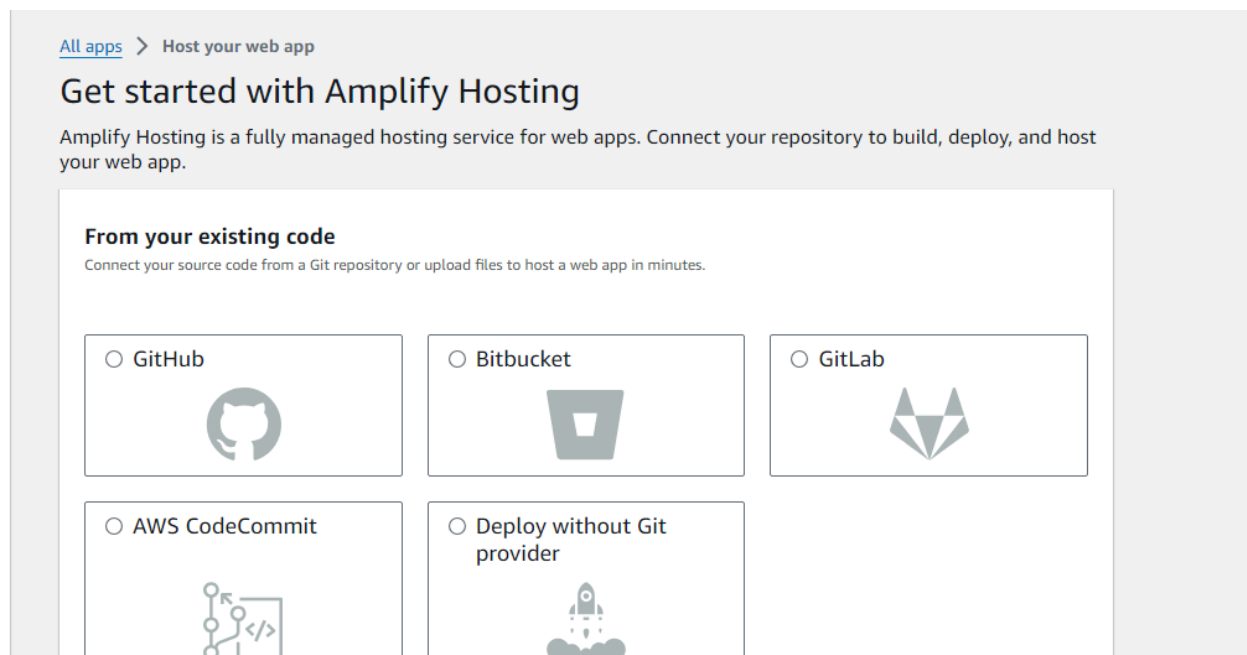
Link to the repository: <https://github.com/TinaHN123/Cloud-Computing-Theory-pro>

Step 4: Connecting source code and uploading web application files from our Github Repository

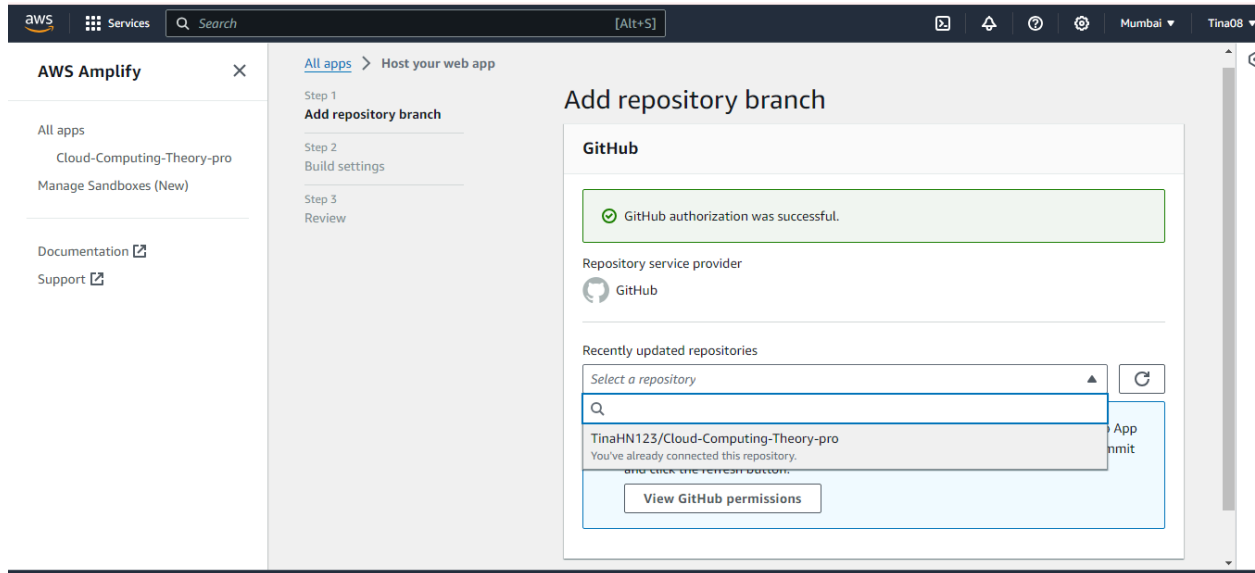
Amazon Amplify Homepage:



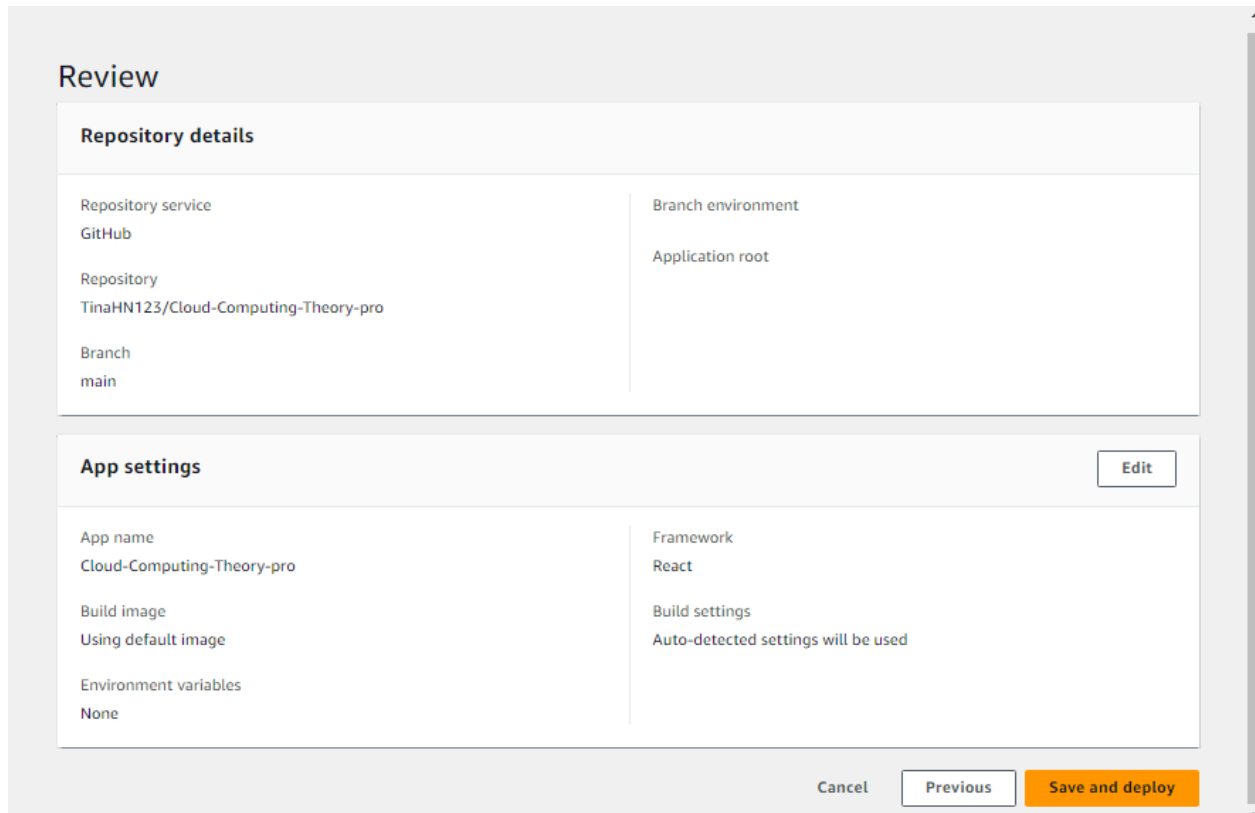
Connecting to Github repository:



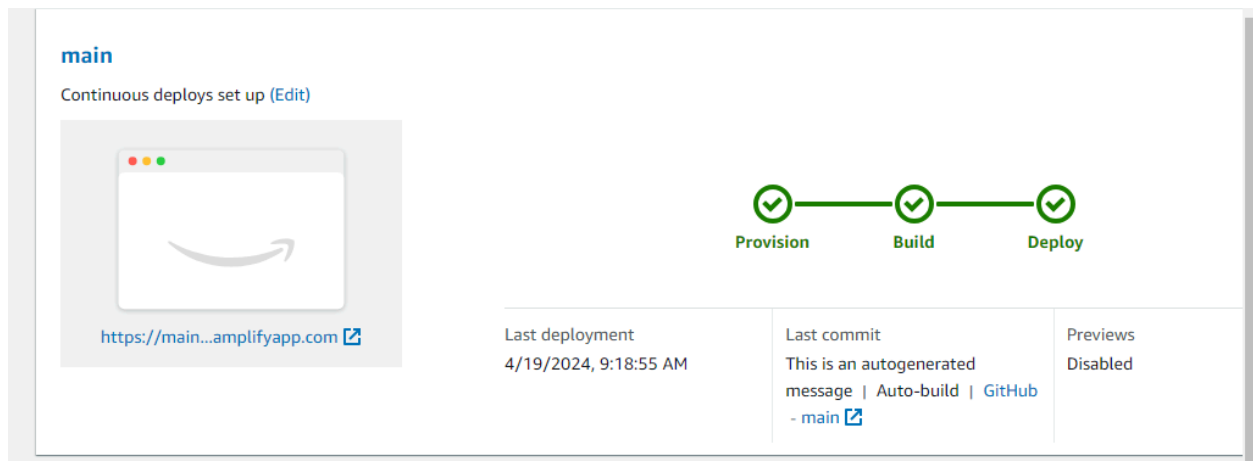
Step 5: Authorization of Github was done after this



Step 6: Review , Save and deploy

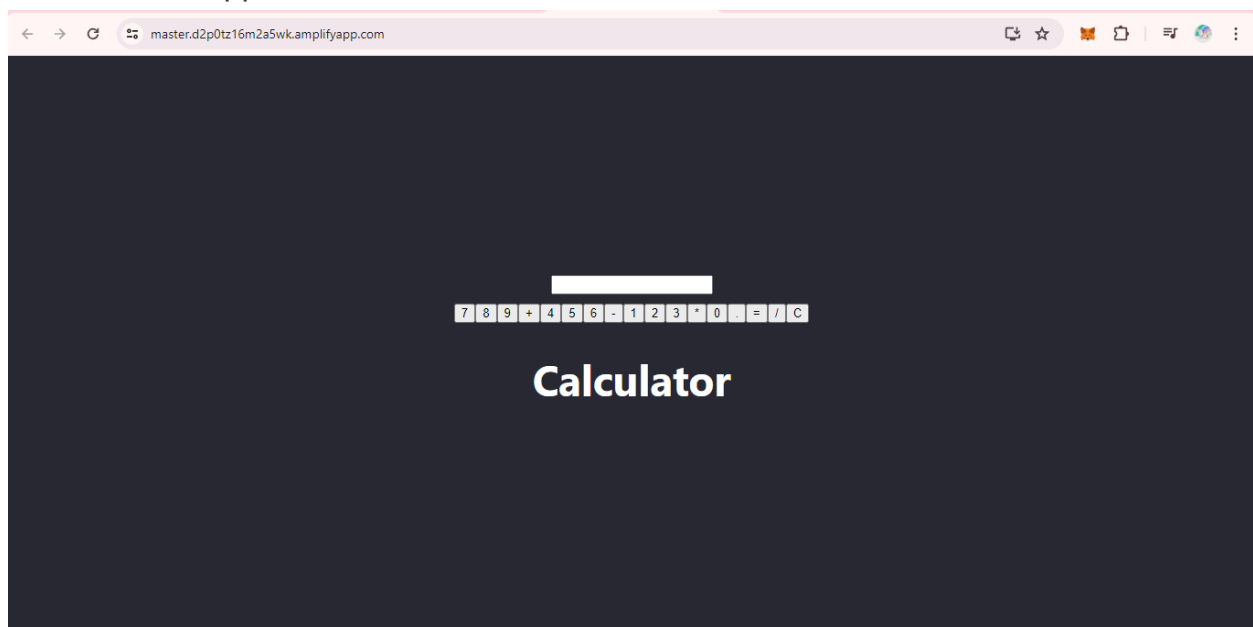


Step 7: The application was deployed on web



Link to webapp: <https://master.d2p0tz16m2a5wk.amplifyapp.com/>

Frontend Webapp:



This is the use case of Platform as a service provided by Amazon AWS Cloud.

Case 2 : Deploying webapp using Amazon S3

Amazon Simple Storage Service (Amazon S3): Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

For an example we deployed a React js web application on Amazon S3

Methodology / Process of deployment:

Step 1: Created a build folder using npm run command on Vscode

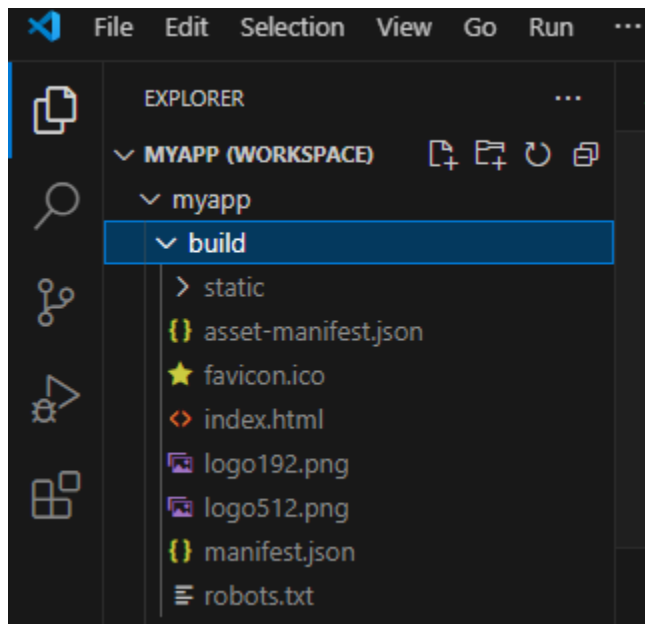
```
PS C:\Users\91916\Desktop\react\myapp> npm run build

> myapp@0.1.0 build
> react-scripts build

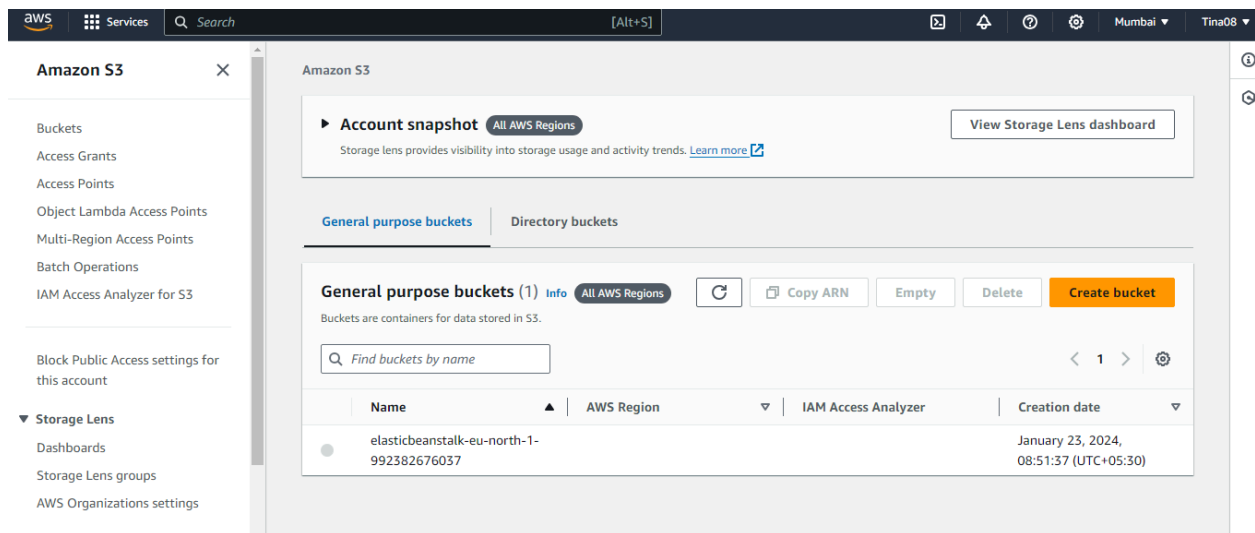
Creating an optimized production build...
One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintained anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Compiled with warnings.
```



Step 2: In AWS S3 bucket , we created a new bucket



Created a Bucket:

aws Services Search [Alt+S]

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name [Info](#)

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

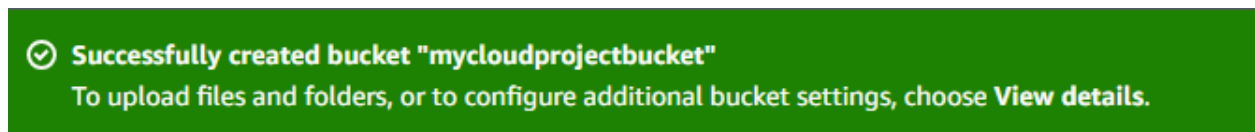
Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Note: Bucket versioning must be enabled

Then the bucket was created:



aws Services Search [Alt+S]

Amazon S3 > Buckets > mycloudprojectbucket

mycloudprojectbucket [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (0) [Info](#)

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

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 permissions. [Learn more](#)

☐ Show versions < 1

Name	Type	Last modified	Size	Storage class
No objects				

You don't have any objects in this bucket.

[Upload](#)

Step 3: Enabled static website hosting in the bucket

[Amazon S3](#) > [Buckets](#) > [mycloudprojectbucket](#) > 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](#)

Step 4: After it was enabled , we got the url , which was used to deploy and host our web app.

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

[Edit](#)

Static website hosting

Enabled

Hosting type

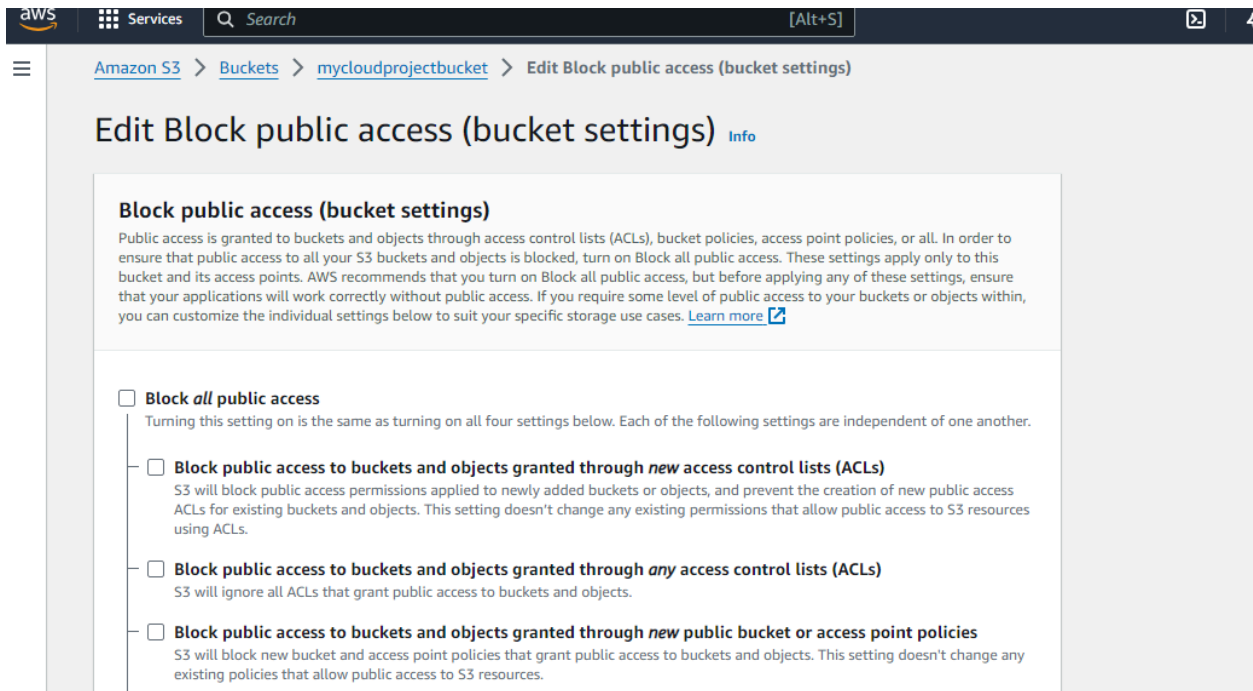
Bucket hosting

Bucket website endpoint

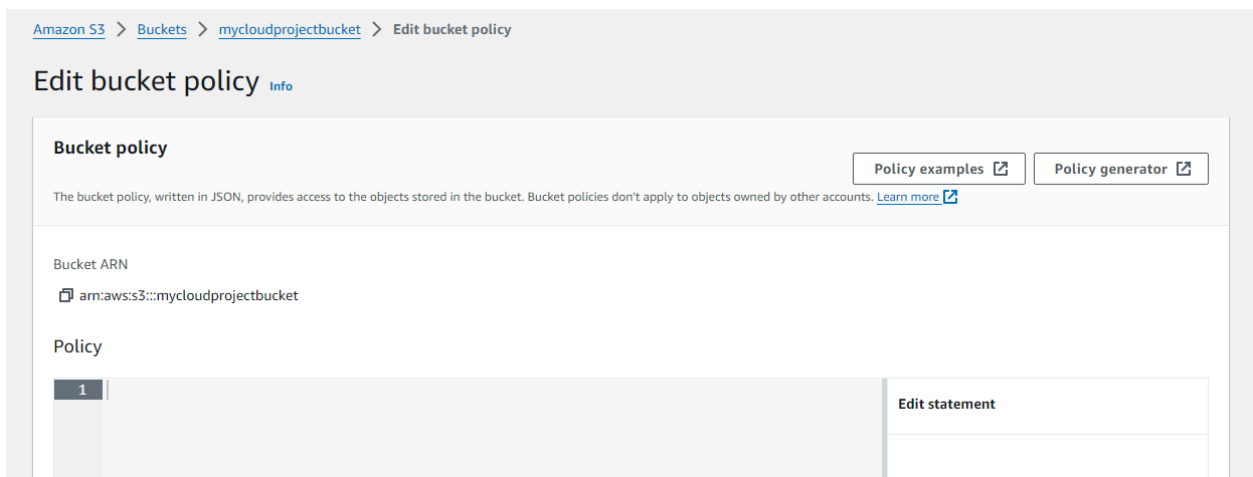
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://mycloudprojectbucket.s3-website-ap-south-1.amazonaws.com>

Step 5: In permissions , Disabled Block Public access to make the web application link publicly accessible



Step 6: To Generate a Bucket policy we used Policy generator



Amazon policy generator:



AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy S3 Bucket Policy ▼

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service Amazon S3 ▼ ☐ All Services ('*')

Use multiple statements to add permissions for more than one service.

Actions 1 Action(s) Selected ▼ ☐ All Actions ('*')

Amazon Resource Name (ARN) arn:aws:s3::mycloudprojec

ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}.
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

[Add Statement](#)

The policy was generated: This is the policy statement

Policy JSON Document

Click below to edit. To save the policy, copy the text below to a text editor.
Changes made below will **not** be reflected in the policy generator tool.

```
{
  "Id": "Policy1713543686176",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1713543678715",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::mycloudprojectbucket",
      "Principal": "*"
    }
  ]
}
```

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind, whether expressed or implied.

[Close](#)

Policy

```
1 {  
2   "Id": "Policy1713543686176",  
3   "Version": "2012-10-17",  
4   "Statement": [  
5     {  
6       "Sid": "Stmt1713543678715",  
7       "Action": [  
8         "s3:GetObject"  
9       ],  
10      "Effect": "Allow",  
11      "Resource": "arn:aws:s3:::mycloudprojectbucket/*",  
12      "Principal": "*"   
13    }  
14  ]  
15 }
```

Step 7: Uploaded files in Objects in bucket from local machine build folder:

<< Desktop > react > myapp > build >

Search build

Name	Date modified	Type
static	19-04-2024 21:40	File folder
asset-manifest	19-04-2024 21:40	JSON Source File
favicon	18-04-2024 21:19	Icon
index	19-04-2024 21:40	Chrome HTML Do
logo192	18-04-2024 21:19	PNG File
logo512	18-04-2024 21:19	PNG File
manifest	18-04-2024 21:19	JSON Source File
robots	18-04-2024 21:19	Text Document

Uploading

33%

Cancel

Total remaining: 2 files: 363.4 KB(66.80%)
Estimated time remaining: a few seconds
Transfer rate: 16.2 KB/s

Upload: status

Close

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

Summary

Destination	Succeeded	Failed
s3://mycloudprojectbucket	🔄 12 files, 180.6 KB (33.20%)	🔄 0 files, 0 B (0%)

Files and folders

Configuration

✔ Upload succeeded
View details below.

Files and folders (14 Total, 544.1 KB)

Name	Folder	Type	Size	Status	Error
favicon.ico	-	image/x-icon	3.8 KB	✔ Succeeded	-
index.html	-	text/html	644.0 B	✔ Succeeded	-
logo192.png	-	image/png	5.2 KB	✔ Succeeded	-
logo512.png	-	image/png	9.4 KB	✔ Succeeded	-
manifest.json	-	application/...	492.0 B	✔ Succeeded	-
robots.txt	-	text/plain	67.0 B	✔ Succeeded	-
main.f855e...	static/css/	text/css	779.0 B	✔ Succeeded	-
main.f855e...	static/css/	-	1.4 KB	✔ Succeeded	-
453.64512f...	static/js/	text/javascript	4.4 KB	✔ Succeeded	-

Step 8: The App was successfully deployed, we can view the frontend using static website link in the bucket:

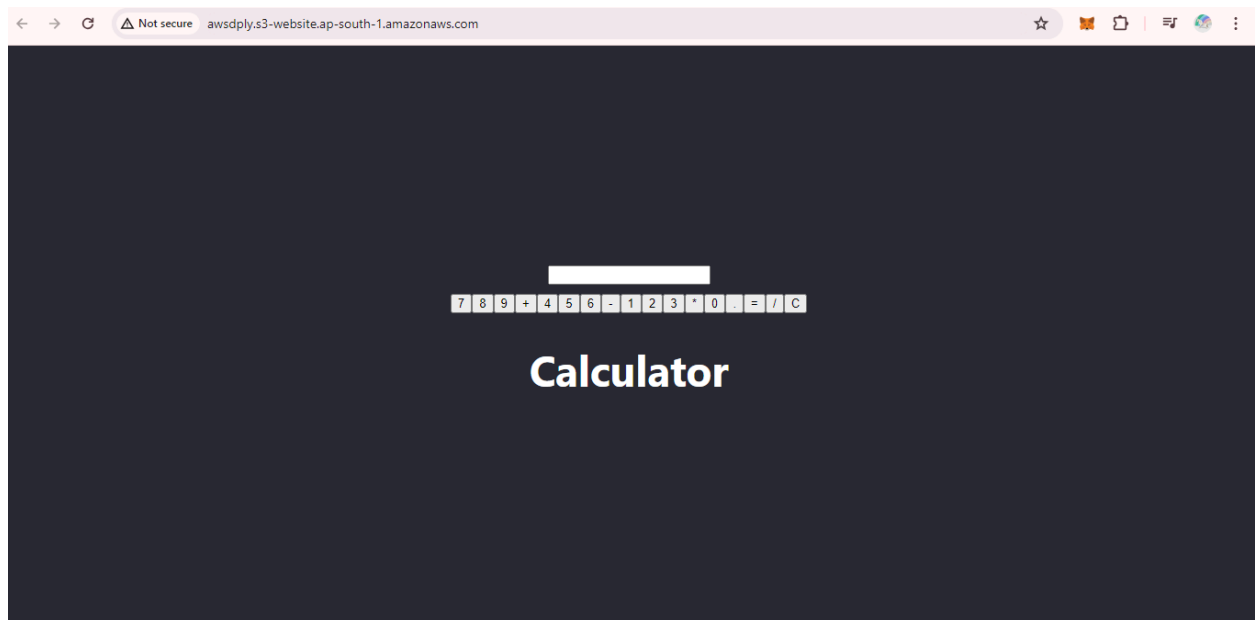
Static website hosting

Edit

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting
Enabled
Hosting type
Bucket hosting
Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)
<http://mycloudprojectbucket.s3-website-ap-south-1.amazonaws.com>

Frontend looks like this:



Link to the website: <http://awsdp1y.s3-website.ap-south-1.amazonaws.com/>

Case 3: Deploying Machine learning model using Sagemaker, S3 bucket and Lambda function

Amazon SageMaker: Amazon SageMaker is a fully managed service that brings together a broad set of tools to enable high-performance, low-cost machine learning (ML) for any use case. With SageMaker, you can build, train and deploy ML models at scale using tools like notebooks, debuggers, profilers, pipelines, MLOps, and more – all in one integrated development environment (IDE). SageMaker supports governance requirements with simplified access control and transparency over your ML projects. In addition, you can build your own FMs, large models that were trained on massive datasets, with purpose-built tools to fine-tune, experiment, retrain, and deploy FMs. SageMaker offers access to hundreds of pretrained models, including publicly available FMs, that you can deploy with just a few clicks.

AWS Lambda: AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it the fastest way to turn an idea into a modern, production, serverless applications.

Both Sagemaker and Lambda are Platform as a Service products.

XGBoost: XGBoost stands for eXtreme Gradient Boosting, and it's an open-source library that provides an efficient and scalable implementation of gradient boosting. Gradient boosting is a machine learning technique used for regression and classification tasks. It is an implementation of gradient boosting decision trees designed for speed and performance. It's an ensemble learning method, which means it combines the predictions of multiple individual models (typically decision trees) to produce a more accurate and robust final prediction. It's particularly well-suited for structured/tabular data and is known for its effectiveness in handling large datasets.

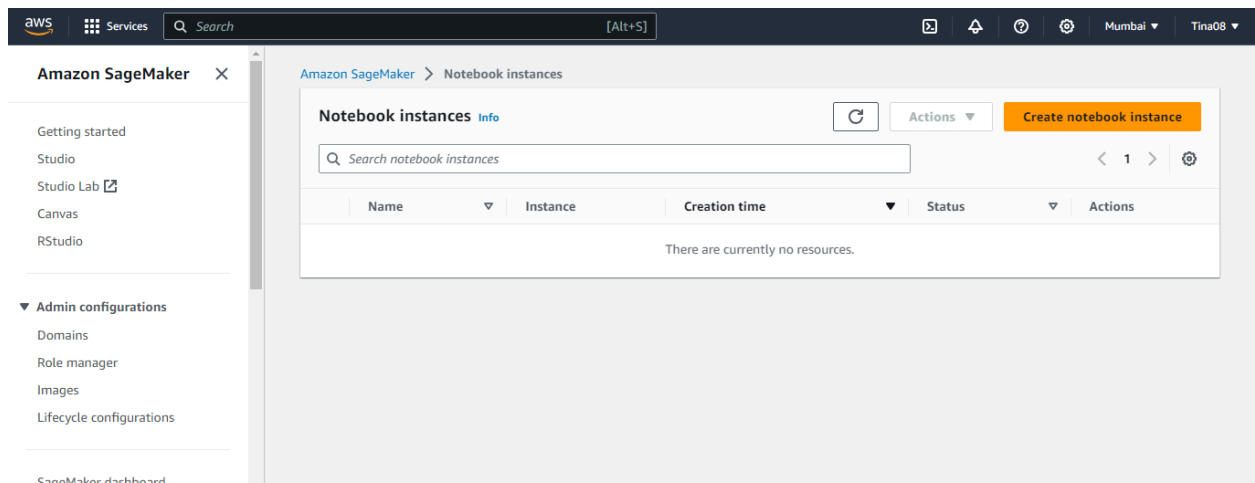
Overview of deployment:

We have deployed our Machine learning XGboost model which is a Gradient boosting algorithm using Sagemaker , S3 bucket for storage and Lambda function to run the code. We created a SageMaker notebook, thereafter creating a SageMaker endpoint, and executing a Lambda function.

Methodology / Process of deployment:

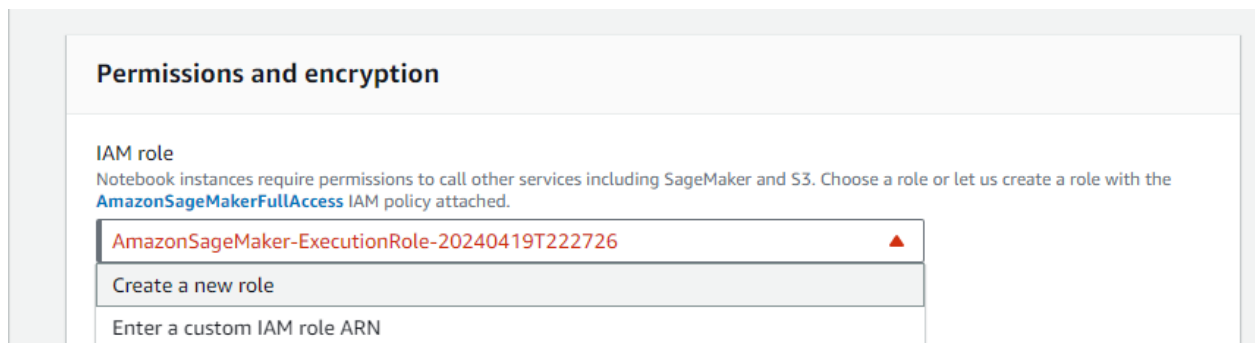
Step 1: Go to Amazon Sagemaker

Step 2: Click on Notebook instance to create notebook instance



Step 3: We created the notebook in order to build and deploy the machine learning model in sagemaker.

Step 4: Created a new role in IAM:



Create an IAM role



Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

☒ S3 buckets you specify - *optional*

☒ Any S3 bucket

Allow users that have access to your notebook instance access to any bucket and its contents in your account.

☐ Specific S3 buckets

Example: bucket-name-1, bucket-name-2, b

Comma delimited. ARNs, "*" and "/" are not supported.

☐ None

☒ Any S3 bucket with "sagemaker" in the name

☒ Any S3 object with "sagemaker" in the name

☒ Any S3 object with the tag "sagemaker" and value "true"

[See Object tagging](#)

☒ S3 bucket with a Bucket Policy allowing access to SageMaker

[See S3 bucket policies](#)

Cancel

Create role

Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20240419T222804 ▼



Success! You created an IAM role.

[AmazonSageMaker-ExecutionRole-20240419T222804](#)



[Create role using the role creation wizard](#)

Root access - *optional*

☒ Enable - Give users root access to the notebook

☐ Disable - Don't give users root access to the notebook

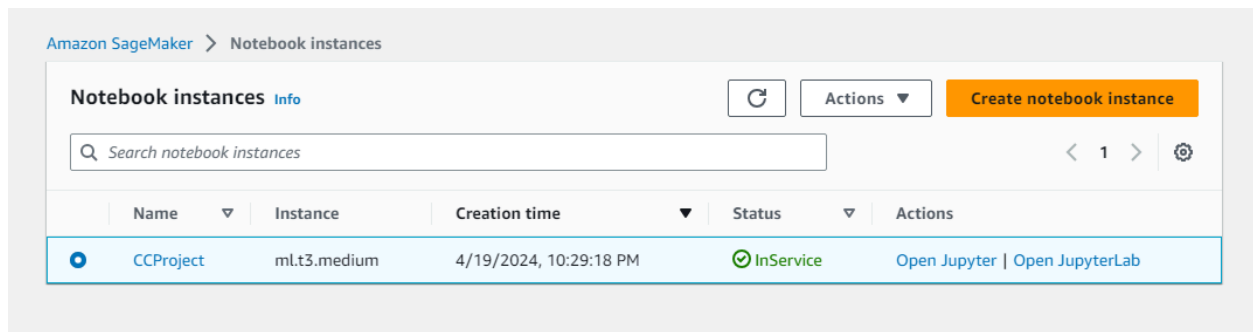
Lifecycle configurations always have root access

Encryption key - *optional*

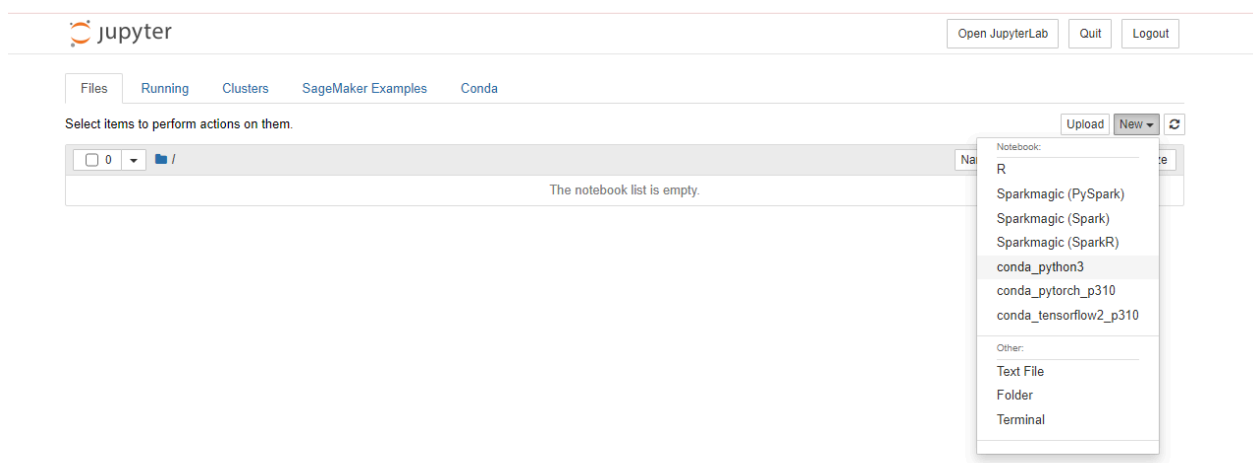
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption ▼

Step 5: A Notebook instance was created. Open Jupyter notebook link redirects us to the jupyter notebook.



Step 6: After opening jupyter notebook : New>Conda python 3



Data preparation

```
In [1]: import urllib.request
        urllib.request.urlretrieve("https://archive.ics.uci.edu/static/public/53/iris.zip", 'data.zip')

Out[1]: ('data.zip', <http.client.HTTPMessage at 0x7f0cc05a3b80>)

In [2]: !mkdir data
        !unzip data.zip -d data/

Archive: data.zip
  inflating: data/Index
  inflating: data/BezdekIris.data
  inflating: data/iris.data
  inflating: data/iris.names
```

After running this code , our data was imported in the notebook.

Select items to perform actions on them.

Upload

New

<input type="checkbox"/>	0		Name	Last Modified	File size
<input type="checkbox"/>		data		2 minutes ago	
<input type="checkbox"/>		Untitled.ipynb	Running	a minute ago	1.73 kB
<input type="checkbox"/>		data.zip		2 minutes ago	3.74 kB

Step 7: Created an S3 Bucket to store the data:

aws

Services

Search

[Alt+S]

Amazon S3

Buckets

Create bucket

Create bucket

Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

Asia Pacific (Mumbai) ap-south-1

Bucket name

Info

sagemaker-build-deploy-model

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

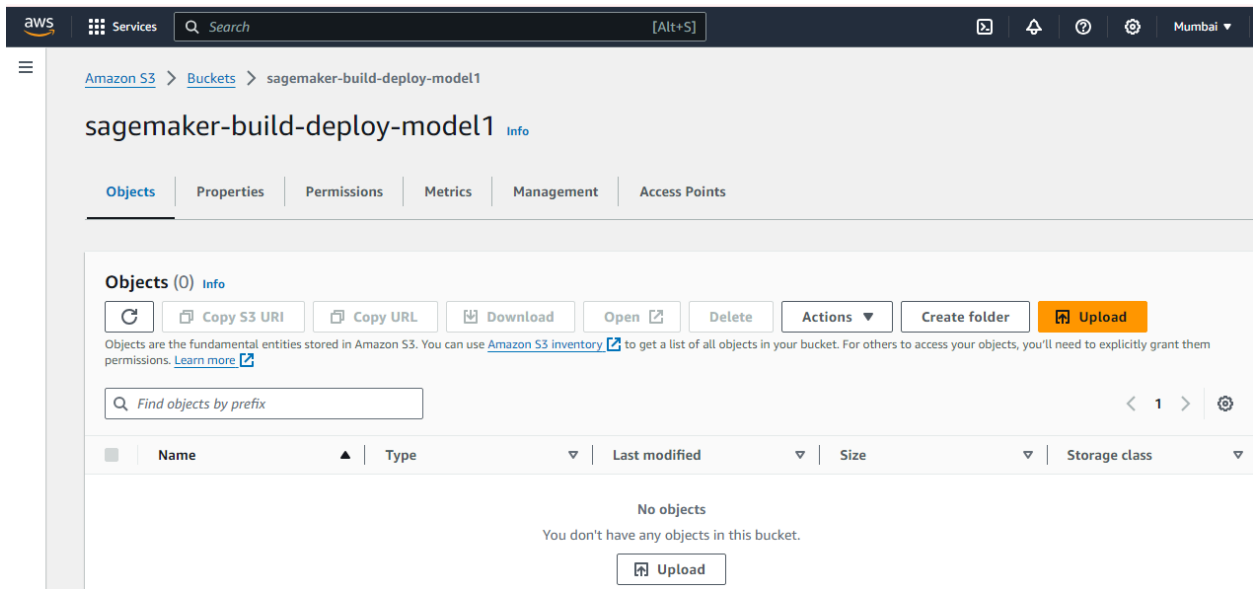
Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Bucket is created



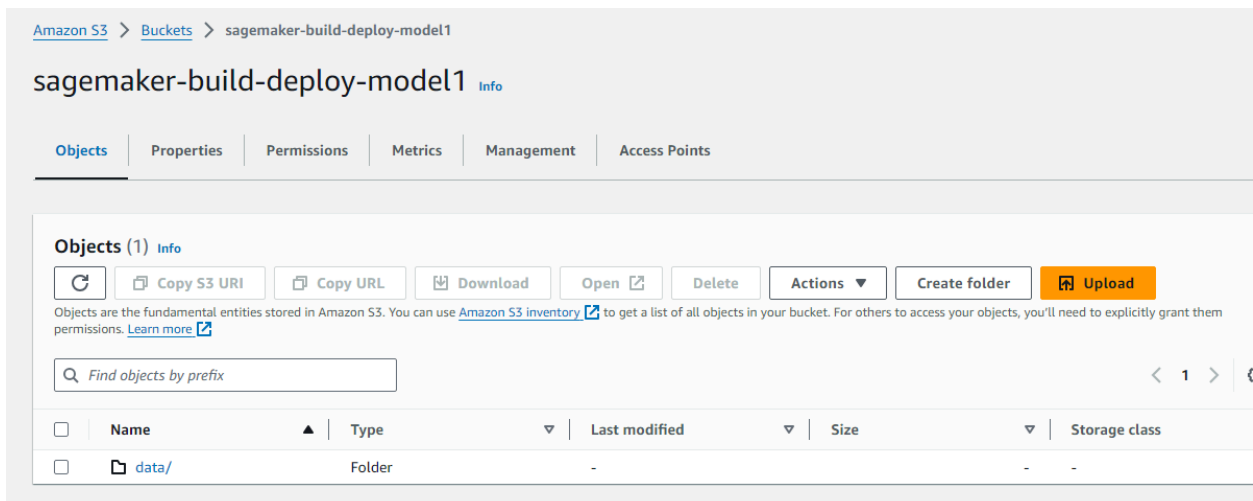
Step 8: The data imported in the notebook will be now stored in this bucket:

Moving Data Into S3 Bucket

```
In [4]: import boto3
bucket_name='sagemaker-build-deploy-model1'

train_data.to_csv('data.csv',header=False,index=False)
key = 'data/train/data'
url = 's3://{}/{}/{}'.format(bucket_name,key)
boto3.Session().resource('s3').Bucket(bucket_name).Object(key).upload_file('data.csv')

val_data.to_csv('data.csv',header=False,index=False)
key = 'data/val/data'
url = 's3://{}/{}/{}'.format(bucket_name,key)
boto3.Session().resource('s3').Bucket(bucket_name).Object(key).upload_file('data.csv')
```



Objects (2) Info

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

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](#)

< 1 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	train/	Folder	-	-	-
<input type="checkbox"/>	val/	Folder	-	-	-

Step 9: Created the Model:

Model Creation

```
|: ▶ import sagemaker
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker import get_execution_role

key='model/xgb_model'
s3_output_location=url='s3://{}/{}'.format(bucket_name,key)

xgb_model=sagemaker.estimator.Estimator(
    get_image_uri(boto3.Session().region_name, 'xgboost'),
    get_execution_role(),
    train_instance_count=1,
    train_instance_type='ml.m4.xlarge',
    train_volume_size=5,
    output_path=s3_output_location,
    sagemaker_session=sagemaker.Session()
)

xgb_model.set_hyperparameters(
    max_depth=5,
    eta=0.2,
    gamma=4,
    min_child_weight=6,
    silent=0,
    objective='multi:softmax',
    num_class=3,
```

Step 10: Then we Trained and Deployed our model inside the notebook:

Training the Model

```
[*]: In train_data='s3://{}/{}/{}'.format(bucket_name,'data/train')
      val_data='s3://{}/{}/{}'.format(bucket_name,'data/val')

      train_channel=sagemaker.session.s3_input(train_data,content_type='text/csv')
      val_channel=sagemaker.session.s3_input(val_data,content_type='text/csv')

      data_channels={'train':train_channel,'validation':val_channel}

      xgb_model.fit(inputs=data_channels)

      The class sagemaker.session.s3_input has been renamed in sagemaker>=2.
      See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
      The class sagemaker.session.s3_input has been renamed in sagemaker>=2.
      See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
      INFO:sagemaker:Creating training-job with name: xgboost-2024-04-19-17-18-31-617

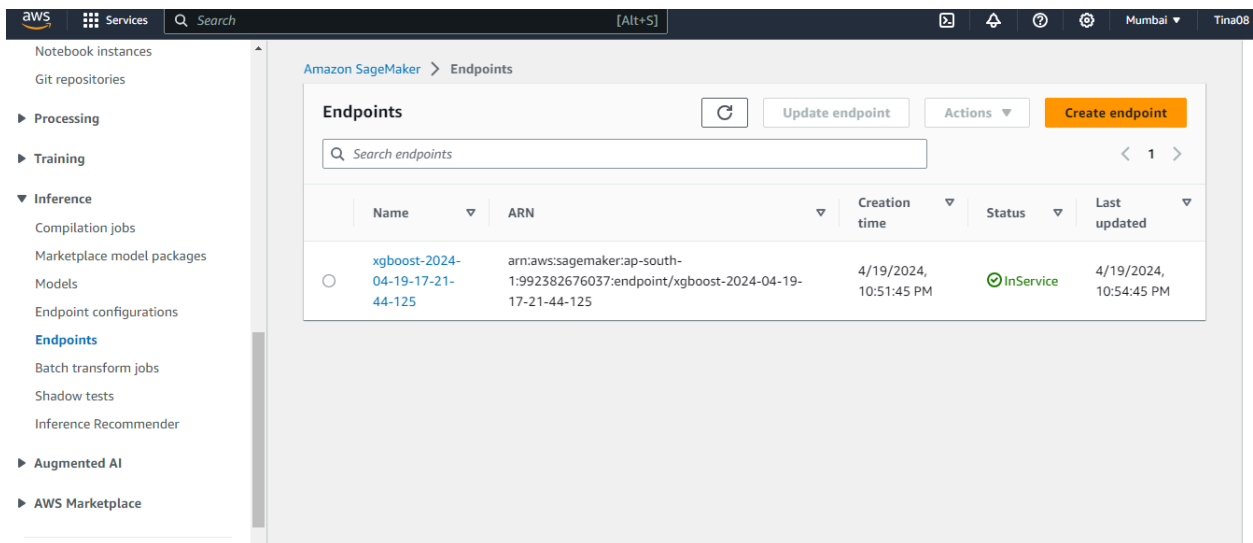
      2024-04-19 17:18:32 Starting - Starting the training job...
      2024-04-19 17:18:47 Starting - Preparing the instances for training.....
```

Deploying the Model

```
[*]: In xgb_predictor=xgb_model.deploy(initial_instance_count=1,
      instance_type='ml.m4.xlarge')
```

Step 11: After Model was deployed we could check the endpoints created in Sagemaker

Click on inference > Click on Endpoints



The screenshot shows the AWS SageMaker console. The left sidebar has a navigation menu with categories like Notebook instances, Git repositories, Processing, Training, Inference, and Augmented AI. The 'Inference' category is expanded, showing options like Compilation jobs, Marketplace model packages, Models, Endpoint configurations, Endpoints, Batch transform jobs, Shadow tests, and Inference Recommender. The 'Endpoints' option is selected. The main panel shows the 'Endpoints' page with a search bar, a table of endpoints, and buttons for 'Update endpoint', 'Actions', and 'Create endpoint'. The table has columns for Name, ARN, Creation time, Status, and Last updated. One endpoint is listed: 'xgboost-2024-04-19-17-21-44-125' with ARN 'arn:aws:sagemaker:ap-south-1:992382676037:endpoint/xgboost-2024-04-19-17-21-44-125', created on 4/19/2024 at 10:51:45 PM, and status 'InService'.

Name	ARN	Creation time	Status	Last updated
xgboost-2024-04-19-17-21-44-125	arn:aws:sagemaker:ap-south-1:992382676037:endpoint/xgboost-2024-04-19-17-21-44-125	4/19/2024, 10:51:45 PM	InService	4/19/2024, 10:54:45 PM

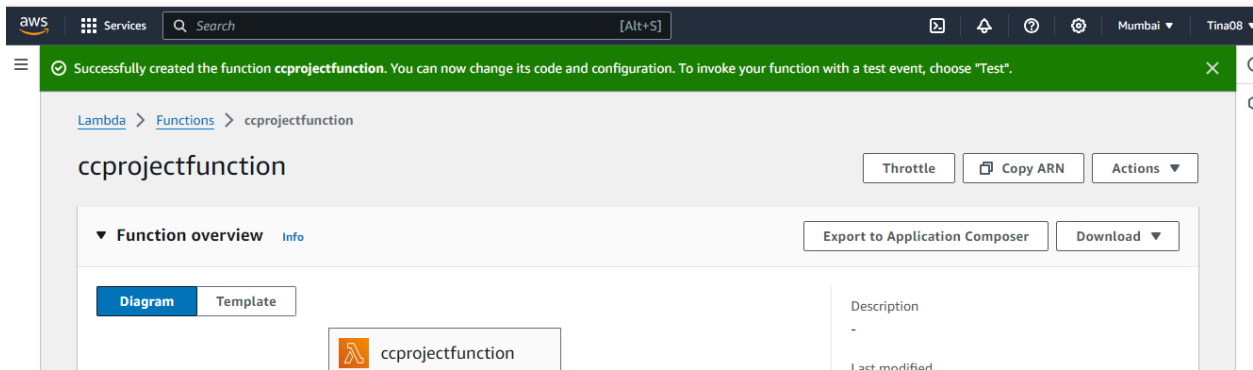
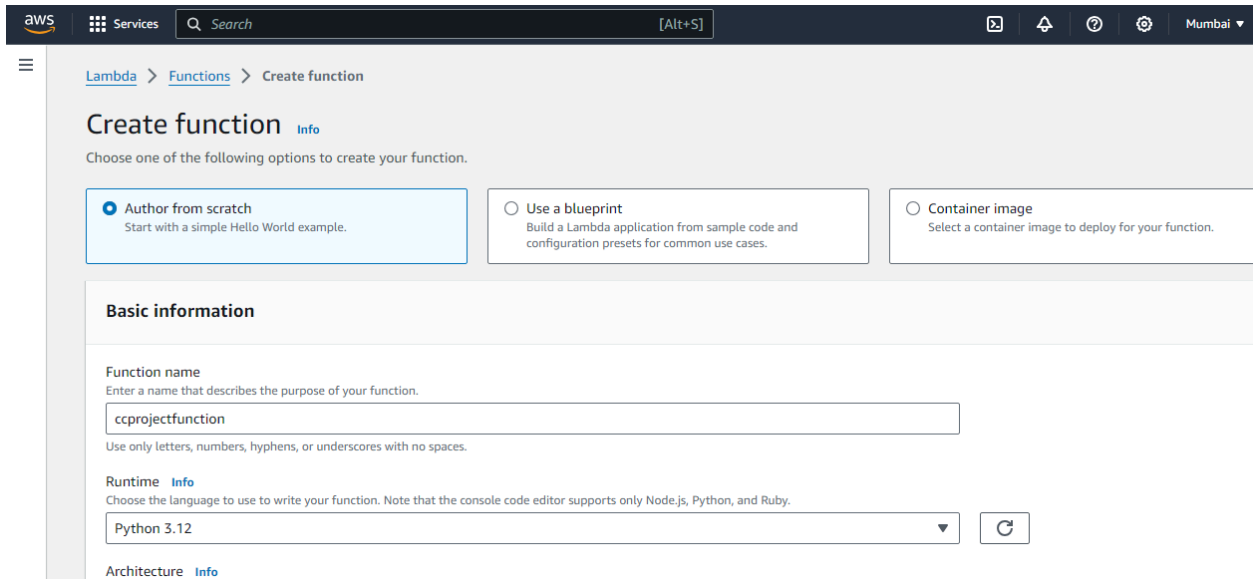
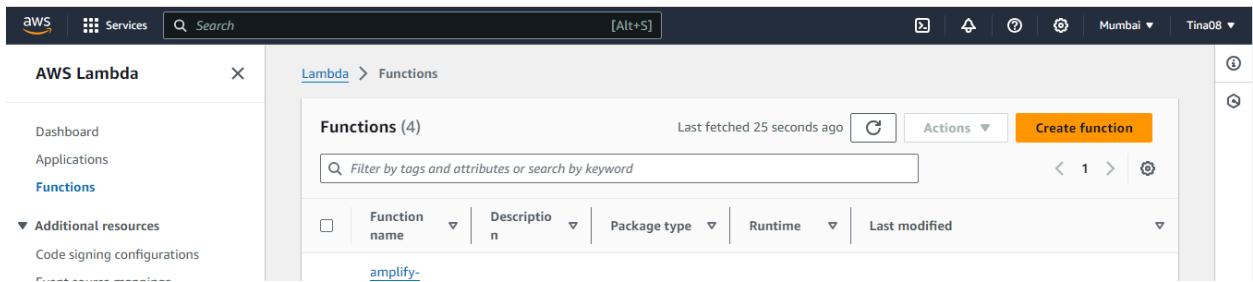
Note: Endpoints were created for predictions

To create an API , we used Lambda function:

Steps are as follows :

Step 12: Created a new function

Lambda Homepage:



Step 13: Then we wrote code for invoking endpoints inside lambda

Step 14: We will first increase the time out period

Lambda > Functions > ccprojectfunction > Edit basic settings

Timeout

1 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/ccprojectfunction-role-qpyb0ua0

[View the ccprojectfunction-role-qpyb0ua0 role](#) on the IAM console.

Step 15: For giving necessary permissions , we created a New role in IAM

Selected lambda as the service

Select trusted entity

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Lambda

Choose a use case for the specified service.
Use case

Add permissions

Permissions policies (1/925)

Choose one or more policies to attach to your new role.



Filter by Type

basic 3 matches

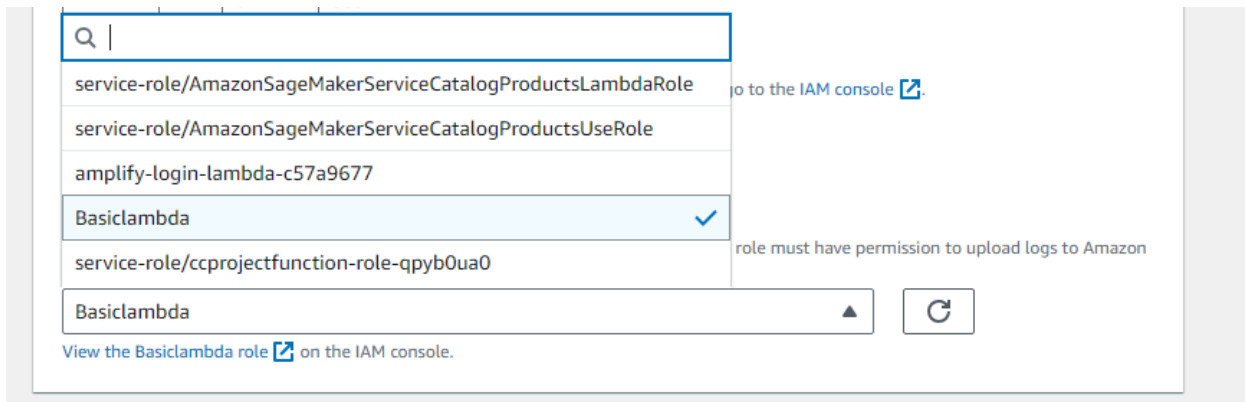
Policy name	Type	Description
<input checked="" type="checkbox"/> AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to CloudWatc...
<input type="checkbox"/> AWSLambdaBasicExecutionRole-17fcab...	Customer managed	-
<input type="checkbox"/> AWSProtonCodeBuildProvisioningBas...	AWS managed	Permissions CodeBuild needs to run a bui...

► Set permissions boundary - optional

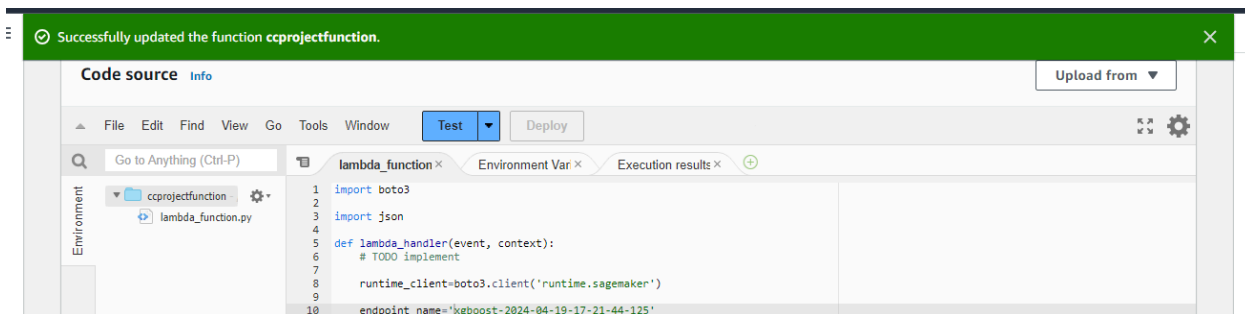
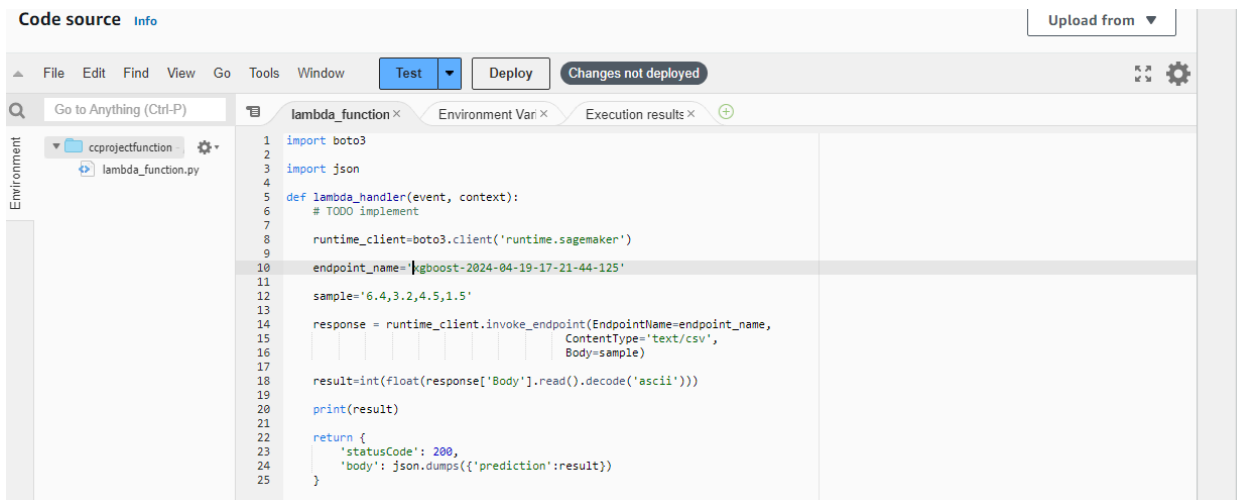
Cancel Previous Next

<input checked="" type="checkbox"/>	 AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to CloudWatc...
<input checked="" type="checkbox"/>	 AmazonSageMakerFullAccess	AWS managed	Provides full access to Amazon SageM...

Step 16: Assigned the role Basiclambda:



Step 17: Then we deployed the code and then tested it inside the Lambda function:



18. Execution result:

Code source Info Upload from ▾

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

- ccprojectfunction
 - lambda_function.py

Execution results

Status: Succeeded Max memory used: 76 MB Time: 2034.96 ms

Test Event Name
(unsaved) test event

Response

```
{
  "statusCode": 200,
  "body": "{\"prediction\": 2}"
}
```

Function Logs

```
START RequestId: ba317e21-3c04-4f0b-a81b-57a33c283c9a Version: $LATEST
2
END RequestId: ba317e21-3c04-4f0b-a81b-57a33c283c9a
REPORT RequestId: ba317e21-3c04-4f0b-a81b-57a33c283c9a Duration: 2034.96 ms Billed Duration: 2035 ms Memory Size: 128 MB Max Memory Used:
```

Request ID
ba317e21-3c04-4f0b-a81b-57a33c283c9a

19. In the code of lambda - function when we manually enter the sample points, from iris dataset , it predicted the correct class

Dataset:

jupyter iris.data✓ 05/23/2023

File Edit View Language

```

107 4.9,2.5,4.5,1.7,Iris-virginica
108 7.3,2.9,6.3,1.8,Iris-virginica
109 6.7,2.5,5.8,1.8,Iris-virginica
110 7.2,3.6,6.1,2.5,Iris-virginica
111 6.5,3.2,5.1,2.0,Iris-virginica
112 6.4,2.7,5.3,1.9,Iris-virginica
113 6.8,3.0,5.5,2.1,Iris-virginica
114 5.7,2.5,5.0,2.0,Iris-virginica
115 5.8,2.8,5.1,2.4,Iris-virginica
116 6.4,3.2,5.3,2.3,Iris-virginica
117 6.5,3.0,5.5,1.8,Iris-virginica
118 7.7,3.8,6.7,2.2,Iris-virginica
119 7.7,2.6,6.9,2.3,Iris-virginica
120 6.0,2.2,5.0,1.5,Iris-virginica
121 6.9,3.2,5.7,2.3,Iris-virginica
122 5.6,2.8,4.9,2.0,Iris-virginica
123 7.7,2.8,6.7,2.0,Iris-virginica
124 6.3,2.7,4.9,1.8,Iris-virginica
125 6.7,3.3,5.7,2.1,Iris-virginica
126 7.2,3.2,6.0,1.8,Iris-virginica
127 6.2,2.8,4.8,1.8,Iris-virginica
128 6.1,3.0,4.9,1.8,Iris-virginica
129 6.4,2.8,5.6,2.1,Iris-virginica
130 7.2,3.0,5.8,1.6,Iris-virginica
131 7.4,2.8,6.1,1.9,Iris-virginica
132 7.9,3.8,6.4,2.0,Iris-virginica
133 6.4,2.8,5.6,2.2,Iris-virginica

```

Sample points:

```
lambda_function x Environment Vari Execution results x (+)
1 import boto3
2
3 import json
4
5 def lambda_handler(event, context):
6     # TODO implement
7
8     runtime_client=boto3.client('runtime.sagemaker')
9
10    endpoint_name='xgboost-2024-04-19-17-21-44-125'
11
12    sample='5.6,2.8,4.9,2.0'
13
14    response = runtime_client.invoke_endpoint(EndpointName=endpoint_name,
15                                              ContentType='text/csv',
16                                              Body=sample)
17
18    result=int(float(response['Body'].read().decode('ascii')))
19
20    print(result)
21
22    return {
23        'statusCode': 200,
24        'body': json.dumps({'prediction':result})
25    }
```


Prediction:

lambda_function x Environment Vari Execution result: x (+)

▼ Execution results Status: Succeeded Max

Test Event Name	(unsaved) test event
Response	{ "statusCode": 200, "body": "{\"prediction\": 1}" }
Function Logs	START RequestId: ba8e03d1-4fbb-4a10-8ff7-928be780668e Version: \$LATEST 1 END RequestId: ba8e03d1-4fbb-4a10-8ff7-928be780668e REPORT RequestId: ba8e03d1-4fbb-4a10-8ff7-928be780668e Duration: 1964.34 ms Billed Duration: 1965 ms Mem
Request ID	ba8e03d1-4fbb-4a10-8ff7-928be780668e

Note : Iris-setosa was assigned the index 0 , iris-virginica was assigned the index as 1 and iris-versicolor as 2. The model correctly predicted the class iris-virginica using the sample points.

```
In [3]:  # read data
import pandas as pd
data=pd.read_csv('data/iris.data',header=None)

# convert to numerical values
data[4]=data[4].replace('Iris-setosa', 0)
data[4]=data[4].replace('Iris-virginica',1)
data[4]=data[4].replace('Iris-versicolor',2)

# shuffle
```

Inference: Storing the dataset in Amazon S3 provided easy access to data for training, validation, and inference. Using Amazon SageMaker for model training offered a scalable and managed environment. SageMaker made it straightforward to deploy trained models as endpoints, allowing for real-time inference. Deploying the XGBoost model as an endpoint enabled us to make predictions on new data with low latency. Creating a Lambda function for predictions enabled us serverless and scalable inference. Lambda functions can be triggered by various events, such as HTTP requests or scheduled events, making it suitable for a wide range of applications. By combining SageMaker, S3, and Lambda, we built an integrated and automated pipeline for machine learning inference. New data can be uploaded to S3, triggering the Lambda function to make predictions using the deployed model. This setup offers flexibility, scalability, and cost-effectiveness for real-world machine learning applications.

References:

<https://docs.aws.amazon.com/sagemaker/>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/tutorials.html>

<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-ex-bucket.html>