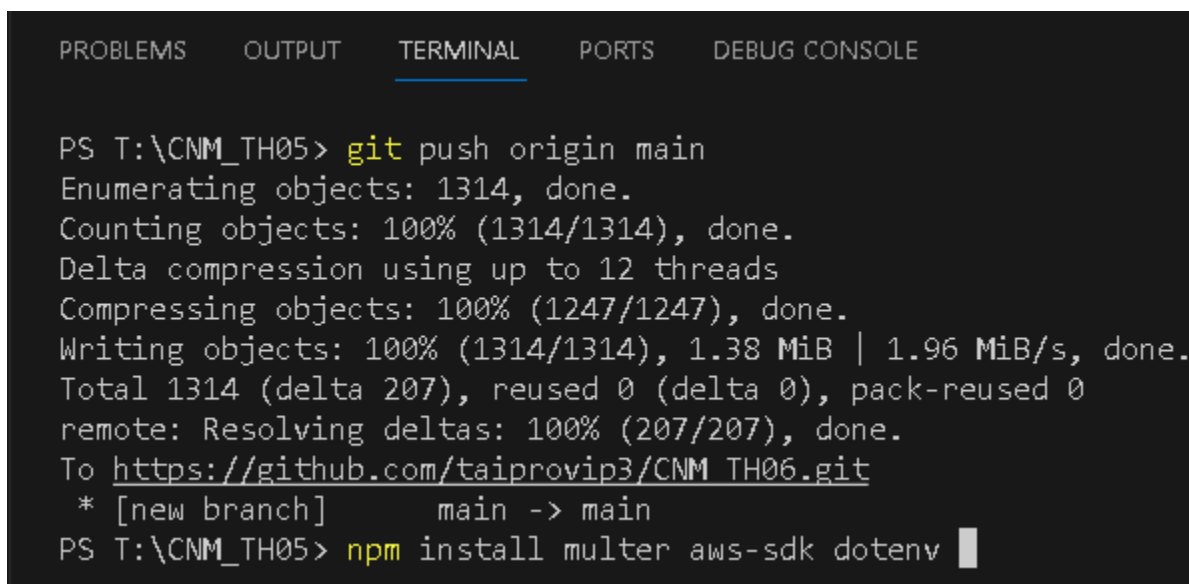


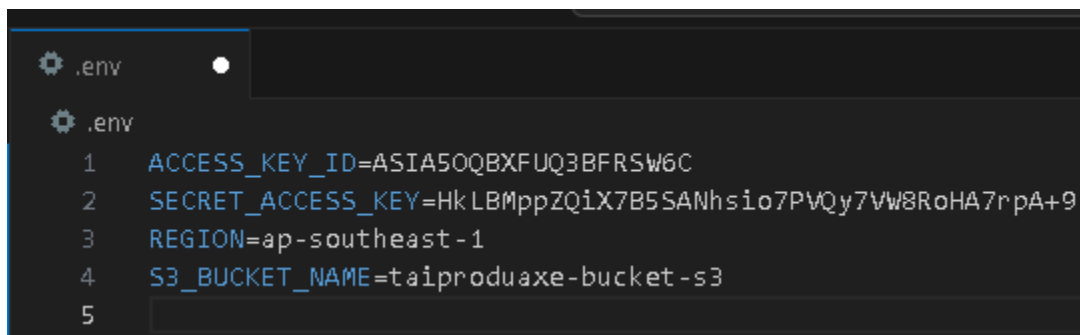
1. Cài đặt thư viện **aws-sdk** và **dotenv** để chuẩn bị kết nối lên Cloud AWS:



```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE

PS T:\CNM_TH05> git push origin main
Enumerating objects: 1314, done.
Counting objects: 100% (1314/1314), done.
Delta compression using up to 12 threads
Compressing objects: 100% (1247/1247), done.
Writing objects: 100% (1314/1314), 1.38 MiB | 1.96 MiB/s, done.
Total 1314 (delta 207), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (207/207), done.
To https://github.com/taiprovip3/CNM_TH06.git
 * [new branch]      main -> main
PS T:\CNM_TH05> npm install multer aws-sdk dotenv
```

2. Tạo file `.env` trong project để lưu trữ các giá trị AccessKey, SecretKey tài khoản IAM User của bạn:



```
.env
1 ACCESS_KEY_ID=ASIA5OQBXFUQ3BFRSW6C
2 SECRET_ACCESS_KEY=HkLBMppZQix7B5SANhsio7PVQy7VW8RoHA7rpA+9
3 REGION=ap-southeast-1
4 S3_BUCKET_NAME=taiproduaxe-bucket-s3
5
```

- Chú ý region **ap-southeast-1** là khu vực của Singapore.

3. Trong trang `index.js`, khai báo thư viện `aws`, `dotenv` và cấu hình AWS kết nối tới Cloud thông qua `accessKey` và `secretAccessKey`.

```

5  const multer = require('multer');
6  const AWS = require('aws-sdk');
7  require('dotenv').config();
8  const path = require('path');
9  |
10 // Cấu hình AWS
11 AWS.config.update({
12   region: process.env.REGION,
13   accessKeyId: process.env.ACCESS_KEY_ID,
14   secretAccessKey: process.env.SECRET_ACCESS_KEY,
15 });
16 const s3 = new AWS.S3();
17 const dynamodb = new AWS.DynamoDB.DocumentClient();
18 const tableName = 'Products';
19 |
20 // Cấu hình App
21 app.use(express.static('./templates'));

```

- **tableName** chính là tên bảng DynamoDB mà bạn đã tạo trên AWS để lưu trữ

4. Tạo thêm input chọn file Image để upload lên S3 trong file index.ejs:

```

<h4>Books:</h4>
<form action="/save" method="post" enctype="multipart/form-data">
  <input type="text" name="maSanPham" id="maSanPham" placeholder="Nhập mã sản phẩm" />
  <br />
  <input type="text" name="tenSanPham" id="tenSanPham" placeholder="Nhập tên sản phẩm" />
  <br />
  <input type="text" name="soLuong" id="soLuong" placeholder="Nhập số lượng" />
  <br />
  <input type="file" name="image" id="image" accept="image/*" />
  <br />
  <input type="submit" value="Thêm" />
</form>

```

- Bổ sung thuộc tính **accept="images/*"** để chỉ cho phép file là ảnh.

5. Cấu hình lại multer để quản lý việc upload hình ảnh:

```

24
25 // Cấu hình multer
26 const storage = multer.memoryStorage({
27   destination(req, file, callback) {
28     callback(null, '');
29   },
30 });
31 const upload = multer({
32   storage,
33   limits: { fileSize: 2000000 }, // Chỉ cho phép file tối đa là 2MB
34   fileFilter(req, file, cb) {
35     checkFileType(file, cb);
36   },
37 });
38 function checkFileType(file, cb) {
39   const fileTypes = /jpeg|jpg|png|gif/;
40
41   const extname = fileTypes.test(path.extname(file.originalname).toLowerCase());
42   const mimetype = fileTypes.test(file.mimetype);
43   if(extname && mimetype) {
44     return cb(null, true);
45   }
46   return cb('Error: Image Only Pls!');
47 }
48

```

- Hàm checkFileType sẽ kiểm tra lại định dạng file upload có phải là hình ảnh hay không.

6. Thay đổi data render trang index.ejs từ mảng thành data lấy từ cloud dynamodb:

```

49 // Routers
50 app.get('/', async (req, res) =>{
51   try {
52     const params = { TableName: tableName };
53     const data = await dynamodb.scan(params).promise();
54     return res.render('index.ejs', { data: data.Items }); // Dùng biến response để render trang 'index.ejs' đồng
    thời truyền biến 'data'
55   } catch (error) {
56     console.error('Error retrieving data from DynamoDB:', error);
57     return res.status(500).send('Internal Server Error');
58   }
59 });
60
61 app.post('/save', upload.single('image'), (req, res) =>{ // Middleware uploadsingl('image') chính định rằng field có

```

7. Lưu dữ liệu 1 item lên Cloud DynamoDB:

```

61 app.post('/save', upload.single('image'), (req, res) => { // Middleware uploads single('image') chính định rằng field có
    name 'image' trong request sẽ được xử lý (lọc, phân tích, kiểm tra dung lượng,..)
62     try {
63         const maSanPham = Number(req.body.maSanPham); // Lấy ra các tham số từ body của form
64         const tenSanPham = req.body.tenSanPham; // Lấy ra các tham số từ body của form
65         const soLuong = Number(req.body.soLuong); // Lấy ra các tham số từ body của form
66
67         const image = req.file.originalname.split('.');
68         const fileType = image[image.length-1];
69         const filePath = `${maSanPham + Date.now().toString()}.${fileType}`; // Custom name của image theo pattern
70
71         const paramsS3 = {
72             Bucket: process.env.S3_BUCKET_NAME,
73             Key: filePath,
74             Body: req.file.buffer,
75             ContentType: req.file.mimetype,
76         }
77         s3.upload(paramsS3, async (err, data) => { // Upload img lên s3 trc
78             if(err) {
79                 console.error('error=', err);
80                 return res.send('Internal server error!');
81             } else { // Sau khi img upload -> sẽ gán URL cho image
82                 const imageURL = data.Location;
83                 console.log('imageURL=', imageURL);
84                 const paramsDynamoDb = {
85                     TableName: tableName,
86                     Item: {

```

```

        s3.upload(paramsS3, async (err, data) => { // Upload img lên s3 trc
        if(err) {
            console.error('error=', err);
            return res.send('Internal server error!');
        } else { // Sau khi img upload -> sẽ gán URL cho image
            const imageURL = data.Location;
            const paramsDynamoDb = {
                TableName: tableName,
                Item: {
                    maSanPham,
                    tenSanPham,
                    soLuong,
                    image: imageURL,
                }
            };
            await dynamodb.put(paramsDynamoDb).promise();
            return res.redirect('/'); // Gọi render lại trang index (để cập nhật dữ liệu table)
        });
    } catch (error) {
        console.error('Error saving data from DynamoDB:', error);
        return res.status(500).send('Internal Server Error');
    }
});

```

- Lưu ý, cần thêm middleware **upload.single('image')** để lọc file là hình ảnh từ form gửi lên trước.

- S3 sẽ trả về urlImage sau khi upload thành công.

8. Xóa item trên cloud DynamoDB:

```

01
02 ✓ app.post('/delete', upload.fields([]), (req, res) => {
03     const listCheckboxSelected = Object.keys(req.body); // Lấy ra tất cả checkboxes
04     // req.body trả về 1 object chứa các cặp key & value định dạng:
05     // '123456': 'on',
06     // '123458': 'on',
07     // listCheckboxSelected trả về 1 array: [ '123456', '123458', '96707133' ]
08 ✓ if(!listCheckboxSelected || listCheckboxSelected.length <= 0){
09     return res.redirect('/');
10 }
11 ✓ try {
12     function onDeleteItem(length){ // Định nghĩa hàm đệ quy xóa
13     const params = {
14         TableName: tableName,
15         Key: {
16             'maSanPham': listCheckboxSelected[length]
17         }
18     }
19     dynamodb.delete(params, (err, data) => {
20     if(err) {
21         console.error('error=', err);
22         return res.send('Internal Server Error!');
23     } else
24         if(length > 0)
25             onDeleteItem(length - 1);
26         else
27             return res.redirect('/');
28     });

```

```

125         onDeleteItem(length - 1);
126     else
127         return res.redirect('/');
128     });
129 }
130 onDeleteItem(listCheckboxSelected.length - 1); // Gọi hàm đệ quy xóa
131 ✓ } catch (error) {
132     console.error('Error deleting data from DynamoDB:', error);
133     return res.status(500).send('Internal Server Error');
134 }
135 });
136
137 ✓ app.listen(4000, () => { // Chạy app ở port 4000

```

- Chạy npm start để xem kết quả