

# Store file into AWS S3 Bucket and access it

IAM User Manu

Bucket name – my-fileprocessor-bucket

Empty bucket before uploading the files.

The screenshot shows the AWS S3 console interface. On the left, a sidebar menu titled 'Amazon S3' lists several sections: 'Buckets', 'Access management and security', 'Storage management and insights', and 'Account and organization settings'. The 'Buckets' section is expanded, showing 'General purpose buckets' with options for 'Directory buckets', 'Table buckets', and 'Vector buckets'. The 'Access management and security' section includes 'Access Points', 'Access Grants', and 'IAM Access Analyzer'. The 'Storage management and insights' section includes 'Storage Lens' and 'Batch Operations'. The main content area is titled 'my-fileprocessor-bucket' and shows the 'Objects' tab selected. It displays a message stating 'No objects' and 'You don't have any objects in this bucket.' There is a prominent 'Upload' button at the bottom. The top navigation bar shows the URL 'ap-south-1.console.aws.amazon.com/s3/buckets/my-fileprocessor-bucket?region=ap-south-1&tab=objects' and the account ID '1525-1738-0311'. The bottom of the page includes links for 'CloudShell', 'Feedback', 'Console Mobile App', and copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

Get File Names –

url – GET -http://localhost:8080/S3/v1/files

204 as no data found

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'pradyesh salunke's Workspace' containing collections like 'Local Storage' and 's3'. The main area shows a 'GET /getFileNames' request to 'http://localhost:8080/S3/v1/files'. The 'Body' tab shows the response: '204 No Content' with a duration of '172 ms' and a size of '112 B'. Below the response, there are tabs for 'Raw', 'Preview', and 'Visualize'.

Now upload the file –

Post - <http://localhost:8080/S3/v1/upload>

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'pradyesh salunke's Workspace' containing collections like 'Local Storage' and 's3'. The main area shows a 'POST /UploadFile1' request to 'http://localhost:8080/S3/v1/upload'. The 'Body' tab shows the request method as 'form-data' and includes a file named 'TestFile1.docx'. The 'Headers' tab shows 'Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0g'. The 'Body' tab also shows the response: '200 OK' with a duration of '1.60 s' and a size of '245 B'. Below the response, there are tabs for 'JSON', 'Preview', and 'Visualize'.

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, environments, history, flows, and files. The main workspace shows a collection named "pradyesh salunke's Workspace" with a "Local Storage" folder containing three POST requests: "UploadFile1", "UploadFile2", and "UploadFile3". Below this is an "s3" folder containing three GET requests: "getFileName1", "getFileName2", and "getFileName3". A specific POST request titled "UploadFile2" is selected, with its details shown in the center panel. The request URL is "http://localhost:8080/S3/v1/upload". The "Body" tab is selected, showing a "form-data" structure with a single field named "file" containing the file "TestFile2.pdf". The "Test Results" tab shows a successful response: "200 OK" with a duration of "307 ms" and a size of "244 B". The response body is a JSON object:

```
1 [ { 2 | "status": "Success", 3 | "detailMessage": "File uploaded successfully::TestFile2.pdf" 4 } ]
```

This screenshot is nearly identical to the one above, showing the same Postman interface and workspace structure. The "UploadFile3" POST request in the "s3" folder is selected. The "Body" tab shows a "form-data" structure with a single file field "file" containing "TestFile3.pdf". The "Test Results" tab shows a successful response: "200 OK" with a duration of "530 ms" and a size of "244 B". The response body is a JSON object:

```
1 [ { 2 | "status": "Success", 3 | "detailMessage": "File uploaded successfully::TestFile3.pdf" 4 } ]
```

## After uploading 3 files to s3 bucket

The screenshot shows the AWS S3 console interface. On the left, there's a navigation sidebar with sections like 'Buckets', 'Access management and security', and 'Storage management and insights'. The main area is titled 'my-fileprocessor-bucket info' and has tabs for 'Objects', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is selected, showing a list of three objects:

Name	Type	Last modified	Size	Storage class
TestFile1.docx	docx	November 27, 2025, 02:22:33 (UTC+05:30)	13.0 KB	Standard
TestFile2.pdf	pdf	November 27, 2025, 02:23:23 (UTC+05:30)	31.7 KB	Standard
TestFile3.pdf	pdf	November 27, 2025, 02:24:25 (UTC+05:30)	31.7 KB	Standard

- Get file with URL – bucket is private but configure pre-signed so that URLs can be access and file download

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections and environments. The main area shows a collection named 'pradyush salunke's Workspace' with a 's3' folder containing a 'getFileName' endpoint. The 'getFileName' endpoint is selected, showing a 'GET' request to 'http://localhost:8080/S3/v1/files'. The 'Body' tab shows a JSON response:

```

1  {
2   "totalFilesPresent": 1,
3   "fileNames": [
4     {
5       "fileName": "TestFile1.docx",
6       "url": "https://my-fileprocessor-bucket.s3.ap-south-1.amazonaws.com/TestFile1.docx?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20251126T185623Z&X-Amz-SignedHeaders=host&X-Amz-Expires=9886&X-Amz-Credential=AKTASHAVAROLR0RECYQDQCF20251126N2Fap-south-1%3F5N3QFaws4_request&X-Amz-Signature=4e6a1917bb2199ce3294da1ff9f0fc423b6f1264f63692e8f89e36a524e5e016"
7     }
8   ]
9 }

```

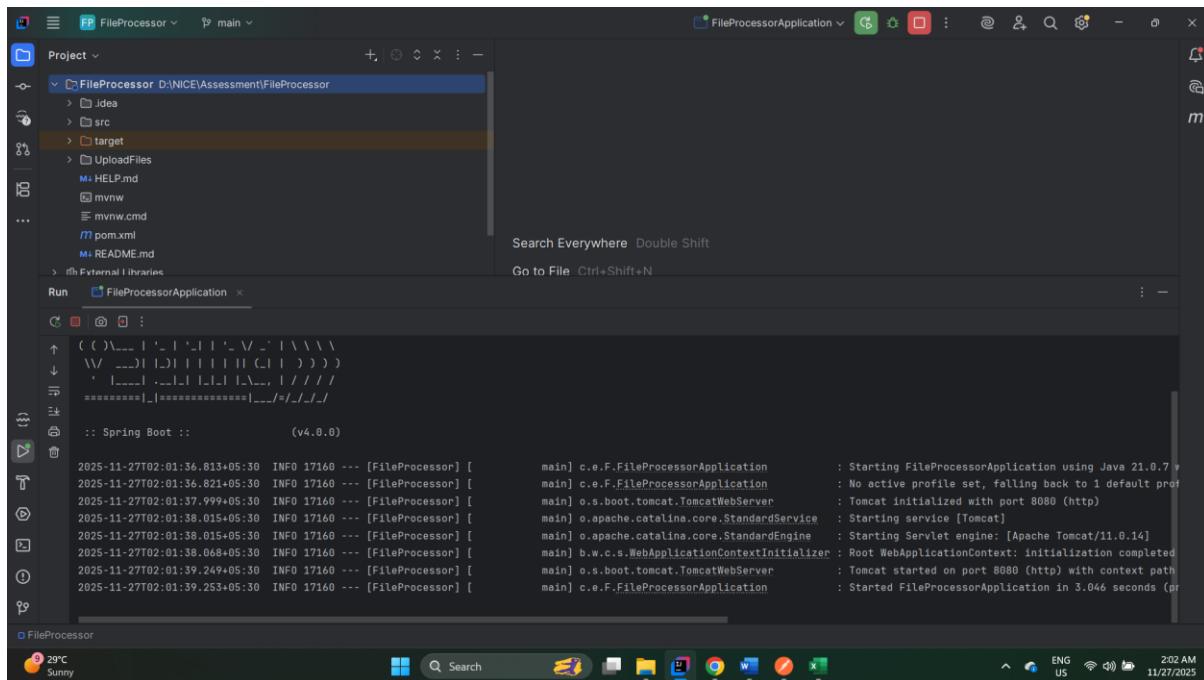
If File is not send then throwing 400 bad request -with message provide correct file

The screenshot shows the Postman interface with a collection named "pradyesh salunke's Workspace". A POST request titled "UploadFile3" is selected. The URL is set to "http://localhost:8080/S3/v1/upload". The "Body" tab is active, showing "form-data" selected. A single field named "file" has its value set to "TestFile3.pdf". The "Headers" tab shows several headers including "Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW". The "Test Results" tab displays the response: "400 Bad Request" with a status of "8 ms" and a size of "328 B". The response body is a JSON object with the following content:

```
1  {
2   "errorCode": "400_BAD_REQUEST",
3   "errorMessage": "Provide correct file",
4   "detailMessage": "org.springframework.web.multipart.MultipartException: Current request is not a multipart request"
5 }
```

# Local Storage Endpoints -

Before local storage endpoint hit-  
there is no folder name files



Testing get file for local – and as expected throw exception

url -GET - http://localhost:8080/local/v1/files

The screenshot shows the Postman interface. On the left, there's a sidebar with collections, environments, history, and flows. The main area shows a collection named "pradyush salunke's Workspace". A specific API endpoint, "getFileNames", is selected. The request URL is "http://localhost:8080/local/v1/files". The response status is "404 Not Found" with a duration of 317 ms and a size of 300 B. The response body is JSON, showing an error message: "errorCode": "404 NOT\_FOUND", "errorMessage": "Folder not found", "detailMessage": "java.io.FileNotFoundException: Folder not found".

## Post – upload -

copied the file into local storage by creating new folder name as files

<http://localhost:8080/local/v1/file/upload>

The screenshot shows the IntelliJ IDEA interface with a project named "FileProcessor" open. The run log window displays the following output:

```

:: Spring Boot ::      (v4.0.0)

2025-11-27T02:01:36.813+05:30 INFO 17160 --- [FileProcessor] [main] c.e.F.FileProcessorApplication : Starting FileProcessorApplication using Java 21.0.7
2025-11-27T02:01:36.821+05:30 INFO 17160 --- [FileProcessor] [main] c.e.F.FileProcessorApplication : No active profile set, falling back to 1 default prof
2025-11-27T02:01:37.999+05:30 INFO 17160 --- [FileProcessor] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-11-27T02:01:38.015+05:30 INFO 17160 --- [FileProcessor] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-11-27T02:01:38.015+05:30 INFO 17160 --- [FileProcessor] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/11.0.14]
2025-11-27T02:01:38.068+05:30 INFO 17160 --- [FileProcessor] [main] b.w.c.s.WebApplicationContextInitializer : Root WebApplicationContext: initialization completed
2025-11-27T02:01:39.249+05:30 INFO 17160 --- [FileProcessor] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path
2025-11-27T02:01:39.253+05:30 INFO 17160 --- [FileProcessor] [main] c.e.F.FileProcessorApplication : Started FileprocessorApplication in 3.046 seconds (pr
2025-11-27T02:03:12.097+05:30 INFO 17160 --- [FileProcessor] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-11-27T02:03:12.097+05:30 INFO 17160 --- [FileProcessor] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-11-27T02:03:12.098+05:30 INFO 17160 --- [FileProcessor] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2025-11-27T02:03:12.166+05:30 ERROR 17160 --- [FileProcessor] [nio-8080-exec-1] c.e.F.e.exceptionController : ERROR -> Folder not found

```

pradyesh salunke's Workspace

Local Storage / UploadFile1

POST http://localhost:8080/local/v1/file/upload

Body (8)  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description
file	TestFile1.docx	

Body Cookies Headers (5) Test Results  JSON  Preview  Visualize

```
1 [ { 2 | "status": "Success", 3 | "detailMessage": "File uploaded successfully::TestFile1.docx" 4 } ]
```

200 OK • 24 ms • 245 B

Save Response

Cloud View Find and replace Console Runner Start Proxy Cookies Vault Trash

pradyesh salunke's Workspace

Local Storage / UploadFile2

POST http://localhost:8080/local/v1/file/upload

Body (8)  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description
file	TestFile2.pdf	

Body Cookies Headers (5) Test Results  JSON  Preview  Visualize

```
1 [ { 2 | "status": "Success", 3 | "detailMessage": "File uploaded successfully::TestFile2.pdf" 4 } ]
```

200 OK • 61 ms • 244 B

Save Response

Cloud View Find and replace Console Runner Start Proxy Cookies Vault Trash

The screenshot shows the Postman interface with a collection named "pradyesh salunke's Workspace". A POST request titled "UploadFile3" is selected, pointing to "http://localhost:8080/local/v1/file/upload". The "Body" tab shows a "form-data" key named "file" with a value of "TestFile3.pdf". The "Test Results" tab displays a successful response with status code 200 OK, execution time of 10 ms, and a content length of 244 B. The response body is JSON, showing a success message: "status: Success" and "detailMessage: File uploaded successfully::TestFile3.pdf".

The screenshot shows the IntelliJ IDEA interface with a project named "FileProcessor". The code editor displays "FileProcessorController.java" with the following code:

```

public class FileProcessorController {
    public FileProcessorController(FileService fileService) { this.fileService = fileService; }

    @PostMapping("file/upload")
    public ResponseEntity<FileResponse> fileUpload(@RequestParam(MultipartFile) MultipartFile file) throws IOException {
        String fileName = fileService.uploadFile(file);
        return new ResponseEntity<FileResponse>(new FileResponse(SUCCESS, detailMessage: FILE_UPLOAD_SUCCESS+SEMI_COLON+fileName));
    }

    @GetMapping("/files")
    public ResponseEntity<List<FileResponse>> getFiles() {
        return new ResponseEntity<List<FileResponse>>(fileService.listFiles());
    }
}

```

The left sidebar shows the project structure with files like "TestFile1.docx", "TestFile2.pdf", and "TestFile3.pdf" in the "files" directory. The bottom right corner shows the terminal output with log messages indicating file uploads to an S3 bucket.

If same file tried to upload again ,will throw file already exist error.

The screenshot shows the Postman interface with a collection named "pradyesh salunke's Workspace". A POST request is being made to `http://localhost:8080/local/v1/file/upload`. The "Body" tab is selected, showing a form-data key "file" with the value "TestFile3.pdf". The response status is 409 Conflict, with the following JSON body:

```
{  
  "errorCode": "409 CONFLICT",  
  "errorMessage": "file already exist at file Path",  
  "detailMessage": "java.nio.file.FileAlreadyExistsException: files\\TestFile3.pdf"  
}
```