Practical: AWS S3 Multi-Part Upload (Manual CLI)**Objective:** To manually execute the S3 Multi-Part Upload process using the AWS Command Line Interface (CLI), breaking a large file into parts, uploading them independently, and then completing the upload for reassembly by S3.

This method is recommended for object sizes greater than **100 MB**.

Phase 1: Prepare the File (Manual Split)

Unlike using an AWS SDK, when using the CLI manually, **Pranav** must first split the file into smaller chunks using a system command like split.

1. **Generate a Large File:** Ensure a large file (e.g., 1.5 GB) exists on your local machine (e.g., order-items-new-data-set.csv).
2. **Split the File:** Use the split command to break the file into parts of a specified size (e.g., 100 MB). This is a Linux/Unix command, not an S3 command.
   Bash
   # Example command to split a file into 100MB parts, naming the output files x00, x01, etc.

```
$ split -b 100m order-items-new-data-set.csv x
# This creates output files like x00, x01, x02...
```

Phase 2: Initiate the Upload and Upload Parts

This phase initiates the process and uploads the individual parts in parallel.

1. **Initiate Multi-Part Upload:** Run the create-multipart-upload command, specifying the S3 bucket and the key (final file name).
   - **Result:** AWS S3 returns a unique **Upload ID**, which is a crucial identifier for all remaining steps.

Bash

```
$ aws s3api create-multipart-upload --bucket pract-5-pranav --key dataset.csv
# Output includes the unique "UploadId"
```

2. **Upload Individual Parts:** For each split file part (e.g., x00, x01), run the upload-part command, specifying the unique metadata for that part.
   - **CRITICAL:** This command must include the **Part Number**, the local file name in the --body parameter, and the **Upload ID**.

Bash
```
# Upload Part 1 (x00)
$ aws s3api upload-part --bucket pract-5-pranav --key dataset.csv --part-number 1 --body xac
--upload-id
WnUruKVhwZNRo.TTNuK6MMOnM0qUD.5Exgrosztlh5ic9zYLTUHoXNCR6omk4c1amNnxZ92
zPcZlcv0ahvHDrnZJU7CCHruo0CQXS2yFn2JqmXOTOFLIzJqX6bHQhIQg

# Upload Part 2 (x01)
$ aws s3api upload-part
  --bucket <your-bucket-name>
  --key order-items-new-data-set.csv
  --part-number 2
  --body x01
  --upload-id <Your-Upload-ID>
# Repeat for all 16 parts
```

3.
    - **Note:** If any part fails, only that part needs to be retried, not the entire upload. Each successful upload returns an **ETag**.

Phase 3: Complete the Upload (Assemble the File)

After all parts are uploaded, you must instruct S3 to assemble them into the final object.

List Parts to Collect ETags: Run the list-parts command to retrieve the list of all successfully uploaded parts, including their required ETag values. The output must be formatted to include only the Part Number and ETag in a JSON format.

Bash
```
# This command uses the required query option to format the output as needed.

$ aws s3api list-parts --bucket pract-5-pranav --key dataset.csv --upload-id
WnUruKVhwZNRo.TTNuK6MMOnM0qUD.5Exgrosztlh5ic9zYLTUHoXNCR6omk4c1amNnxZ92
zPcZlcv0ahvHDrnZJU7CCHruo0CQXS2yFn2JqmXOTOFLIzJqX6bHQhIQg --query '{Parts:
Parts[].{PartNumber:PartNumber,ETag:ETag}}'  > output.json

# The output is saved to output.json
```
Complete the Upload: Use the complete-multipart-upload command, referencing the output.json file to tell S3 which parts to combine.

Replace output.json with
```
{
  "Parts": [
    {
```

```
    "PartNumber": 1,
    "ETag": "\"etag1\""
   },
   {
    "PartNumber": 2,
    "ETag": "\"etag2\""
   },
   {
    "PartNumber": 3,
    "ETag": "\"etag3\""
   }
  ]
}
}
```

Bash

```
$ aws s3api complete-multipart-upload --bucket pract-5-pranav --key dataset.csv --upload-id WnUruKVhwZNRo.TTNuK6MMOnM0qUD.5Exgrosztlh5ic9zYLTUHoXNCR6omk4c1amNnxZ92zPcZlcv0ahvHDrnZJU7CCHruo0CQXS2yFn2JqmXOTOFLIzJqX6bHQhIQg --multipart-upload file://output.json
```

1.
    - **Validation:** The upload is now complete, and the final 1.5 GB file appears in the S3 bucket.

Phase 4: Cleanup (Abort Non-Completed Uploads)

If an upload fails or is canceled before completion, you must explicitly abort it to avoid storage charges for orphaned parts.

Abort Command: Use the abort-multipart-upload command with the Upload ID of the non-completed upload.

Bash

```
$ aws s3api abort-multipart-upload \
  --bucket <your-bucket-name> \
  --key <Non-Completed-File-Key> \
  --upload-id <Non-Completed-Upload-ID>
```

1.
    - This action removes all parts associated with that Upload ID.