# Week 10 — S3 → Lambda Event-Driven Upload Trigger

**What this document contains**

* A complete, step-by-step guide that shows every configuration you’ll need (console + CLI + IAM policies + Lambda code).
* A ready-to-use \*\*GitHub \*\*`` you can drop into a repo.
* A polished, engaging **blog article** you can publish in your portfolio that showcases the project and points where you should place screenshots.

**Architecture GIF placeholder:** sandbox:/mnt/data/architecture.gif

## Quick project summary

When a file is uploaded to an Amazon S3 bucket, that S3 event triggers an AWS Lambda function. The Lambda function will:

* Log the event (bucket, key, time) to CloudWatch Logs (primary behavior), and
* Optionally read object metadata (head\_object) and optionally send an email using Amazon SES.

This document walks you through creating the S3 bucket, creating a Lambda execution role with the required IAM policies, deploying the Lambda (console or CLI), wiring up S3 event notifications, testing everything, and preparing screenshots and deliverables.

# Table of contents

1. Prerequisites
2. Architecture overview (place GIF)
3. Step-by-step (console + CLI) — full details
   * Create S3 bucket
   * Create IAM role for Lambda
   * Create Lambda function (code)
   * Add S3 trigger to Lambda
   * Test & verify
   * Optional: Add SES email notifications
4. Complete IAM policy JSONs (copy/paste)
5. Lambda code (complete)
6. CloudFormation template (optional; one-file deploy)
7. CLI quick deploy commands (copy/paste)
8. Troubleshooting & debugging
9. Deliverable checklist (what to screenshot)
10. Cleanup commands

## 1) Prerequisites

* An AWS account.
* One region where you’ll operate (I use us-east-1 in examples); keep S3, Lambda, and SES in the same region.
* AWS CLI installed & configured (aws configure with your credentials), if you want to run CLI commands.
* Basic familiarity with the AWS Console (we provide console steps for each item).

## 2) Architecture overview

Insert the animated GIF showing the flow here in your README and blog. The GIF file we generated is at: sandbox:/mnt/data/architecture.gif.

**Where to place screenshots in README & blog:**

* S3 bucket overview/properties screenshot.
* Lambda function configuration (role, runtime, handler) screenshot.
* Lambda function code screenshot (showing the handler).
* Lambda Designer showing the S3 trigger connected to the function (or S3 Event Notifications page).
* CloudWatch log entry showing the object key and timestamp.
* If using SES: screenshot of SES verified identities or received email.

Placeholders in this doc use <!-- SCREENSHOT: name --> so you can quickly search and replace.

## 3) Step-by-step (console + exact CLI commands)

### A — Create an S3 bucket

**Console**

1. AWS Console → Services → S3 → **Create bucket**.
2. Bucket name: my-week10-bucket-<your-unique-suffix> (MUST be globally unique).
3. Region: choose your desired region (example uses us-east-1).
4. Keep **Block Public Access** ON (recommended).
5. Optionally enable default encryption (AWS‑managed) in **Properties** → **Default encryption** → Enable → SSE‑S3.
6. Click **Create bucket**.

**CLI**

aws s3api create-bucket --bucket my-week10-bucket-12345 --region us-east-1 \  
 --create-bucket-configuration LocationConstraint=us-east-1  
# (Note: for us-east-1 you may omit LocationConstraint)

### B — Create IAM role for Lambda (with CloudWatch, and optional S3/SES permissions)

We will create a role that trusts Lambda and attach the AWS-managed AWSLambdaBasicExecutionRole for CloudWatch logging. If Lambda must read objects (head\_object) or call SES, attach a custom policy with explicit resources.

**Trust policy (Lambda)**

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Effect": "Allow",  
 "Principal": { "Service": "lambda.amazonaws.com" },  
 "Action": "sts:AssumeRole"  
 }  
 ]  
}

**Console**

1. Console → IAM → Roles → **Create role**.
2. Trusted entity: **AWS service** → **Lambda** → Next.
3. Attach AWSLambdaBasicExecutionRole.
4. (Optional) Create and attach a custom policy for S3 read and/or SES send actions (see policy JSONs below).
5. Name the role: lambda-s3-exec-role-week10.

**CLI (create role + attach)**

# create role  
aws iam create-role --role-name lambda-s3-exec-role-week10 \  
 --assume-role-policy-document file://trust.json  
# attach AWS managed policy for CloudWatch logs  
aws iam attach-role-policy --role-name lambda-s3-exec-role-week10 \  
 --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

### C — Create the Lambda function

We include the exact code below (Section 5). Create a function with Python runtime.

**Console**

1. Console → Lambda → **Create function** → Author from scratch.
2. Function name: s3-upload-trigger-week10.
3. Runtime: Python 3.9 (or latest available).
4. Permissions: choose **Use an existing role** → select lambda-s3-exec-role-week10.
5. Create function.
6. In **Configuration**: set Timeout 30s, Memory 128MB.
7. Add environment variables (optional):
   * SEND\_EMAIL = false
   * SES\_SOURCE = verified-sender@example.com
   * SES\_DEST = verified-recipient@example.com
8. Paste the Lambda code (see Section 5) into the inline editor and Deploy.

**CLI (zip & create)**

zip function.zip lambda\_function.py  
aws lambda create-function --function-name s3-upload-trigger-week10 \  
 --runtime python3.9 --role arn:aws:iam::123456789012:role/lambda-s3-exec-role-week10 \  
 --handler lambda\_function.lambda\_handler --zip-file fileb://function.zip --timeout 30 --memory-size 128

### D — Add S3 trigger (two ways)

**Method A — From Lambda console (recommended)**

1. Open your Lambda function → **Add trigger**.
2. Choose **S3**.
3. Bucket: my-week10-bucket-...
4. Event type: All object create events (or PUT only).
5. Prefix / Suffix (optional).
6. Add. Lambda console will auto-add permission so S3 can invoke Lambda.

**Method B — From S3 console**

1. S3 → select bucket → **Properties** → **Event notifications** (or **Events** tab).
2. **Create event notification**: select events PUT or All object create events, Destination: **Lambda function**, and pick the function.

**CLI** (manual notification + add permission)

aws lambda add-permission --function-name s3-upload-trigger-week10 \  
 --statement-id s3invoke --action "lambda:InvokeFunction" --principal s3.amazonaws.com \  
 --source-arn arn:aws:s3:::my-week10-bucket-12345  
  
aws s3api put-bucket-notification-configuration --bucket my-week10-bucket-12345 --notification-configuration '{  
 "LambdaFunctionConfigurations": [  
 {  
 "LambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:s3-upload-trigger-week10",  
 "Events": ["s3:ObjectCreated:\*"]  
 }  
 ]  
}'

### E — Test & verify

1. Upload a file to your S3 bucket (Console or CLI):

* echo "hello week10" > test-file.txt  
  aws s3 cp test-file.txt s3://my-week10-bucket-12345/test-file.txt

1. Open CloudWatch Logs → Log groups → /aws/lambda/s3-upload-trigger-week10 → open the latest log stream.
   * Look for messages like: New object in bucket 'my-week10-bucket-12345': key='test-file.txt'

## 4) IAM policy JSONs (copy/paste)

**S3 read (if Lambda will call** ``**)**

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "AllowS3ReadForSpecificBucket",  
 "Effect": "Allow",  
 "Action": ["s3:GetObject","s3:ListBucket","s3:GetObjectAcl"],  
 "Resource": [  
 "arn:aws:s3:::my-week10-bucket-12345",  
 "arn:aws:s3:::my-week10-bucket-12345/\*"  
 ]  
 }  
 ]  
}

**SES send (optional)**

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {  
 "Effect":"Allow",  
 "Action":["ses:SendEmail","ses:SendRawEmail"],  
 "Resource":"\*"  
 }  
 ]  
}

Note: SES often requires you to verify the sender identity, and SES accounts may be in sandbox mode (recipient must be verified until you request production access).

## 5) Lambda function code (complete)

Create lambda\_function.py with the following content and deploy to your Lambda. This function logs the S3 event, optionally reads object metadata, and optionally sends an SES email when environment variable SEND\_EMAIL=true.

import json  
import os  
import logging  
import urllib.parse  
import boto3  
  
logger = logging.getLogger()  
logger.setLevel(logging.INFO)  
  
s3\_client = boto3.client('s3')  
ses\_client = boto3.client('ses') # only used if SEND\_EMAIL = true  
  
  
def lambda\_handler(event, context):  
 logger.info("Received event: %s", json.dumps(event))  
  
 records = event.get('Records', [])  
 for record in records:  
 try:  
 s3 = record['s3']  
 bucket = s3['bucket']['name']  
 key = urllib.parse.unquote\_plus(s3['object']['key'])  
 logger.info("New object in bucket '%s': key='%s'", bucket, key)  
  
 # Optional: read metadata (head\_object)  
 try:  
 head = s3\_client.head\_object(Bucket=bucket, Key=key)  
 logger.info("Object metadata: ContentLength=%s ContentType=%s",  
 head.get('ContentLength'), head.get('ContentType'))  
 except Exception:  
 logger.exception("Could not read object head (maybe no permission).")  
  
 # Optional SES email  
 send\_email = os.environ.get('SEND\_EMAIL', 'false').lower() == 'true'  
 if send\_email:  
 source = os.environ.get('SES\_SOURCE')  
 dest = os.environ.get('SES\_DEST')  
 if not source or not dest:  
 logger.error("SES\_SOURCE or SES\_DEST environment variables not set.")  
 else:  
 subject = f"New upload: {key}"  
 body = f"A new object was uploaded to {bucket}/{key}."  
 try:  
 ses\_client.send\_email(  
 Source=source,  
 Destination={'ToAddresses': [dest]},  
 Message={  
 'Subject': {'Data': subject},  
 'Body': {'Text': {'Data': body}}