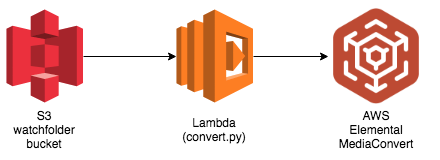
# **Automating MediaConvert Jobs with Lambda**

We'll use Amazon S3 events and Lambda to automatically trigger AWS Elemental MediaConvert jobs. The ability to watch S3 folders and take action on incoming items is a useful automation technique that enables the creation of fully unattended workflows. In our case, the user will place videos in a folder on AWS S3 which will trigger an ingest workflow to convert the video. We'll use the job definition from the previous modules as the basis for our automated job.



You'll implement a Lambda function that will be invoked each time a user uploads a video. The lambda will send the video to the MediaConvert service to produce several outputs:

* An Apple HLS adaptive bitrate stream for playout on multiple sized devices and varying bandwidths.
* An MP4 stream
* Thumbnails for use in websites to show a preview of the video at a point in time.

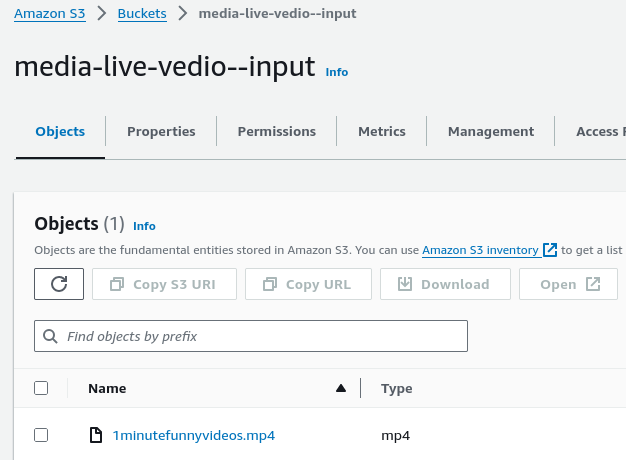
Converted outputs will be saved in the S3 MediaBucket created earlier in the lab.

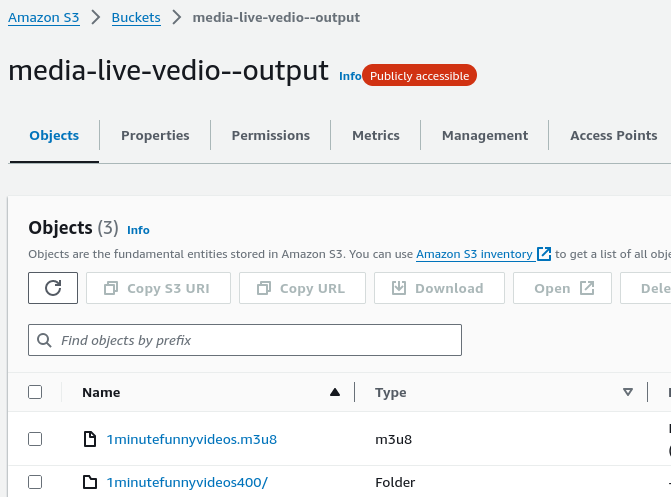
## **Implementation Instructions**

### [Create an Amazon S3 bucket to use for uploading videos to be converted](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/README.md#watchfolder-bucket)

Use the console or AWS CLI to create an Amazon S3 bucket.

1. In the AWS Management Console choose Services then select S3 under Storage.
2. Choose +Create Bucket
3. Provide a globally unique name for your bucket
4. Select the Region you've chosen to use for this workshop from the dropdown.
5. For both output and input vedios buckets





### [2. Create an IAM Role for Your Lambda function](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/README.md#vod-lambda-role) :

1. Every Lambda function has an IAM role associated with it. This role defines what other AWS services the function is allowed to interact with. For the purposes of this workshop, you'll need to create an IAM role that grants your Lambda function permission to interact with the MediaConvert service
2. Step-by-step instructions
3. From the AWS Management Console, click on Services and then select IAM in the Security, Identity & Compliance section.
4. Select Roles in the left navigation bar and then choose Create role.
5. Select AWS Service and Lambda for the role type, then click on the Next:Permissions button.
6. Note: Selecting a role type automatically creates a trust policy for your role that allows AWS services to assume this role on your behalf. If you were creating this role using the CLI, AWS CloudFormation or another mechanism, you would specify a trust policy directly.
7. Begin typing AWSLambdaBasicExecutionRole in the Filter text box and check the box next to that role.
8. Delete what you entered in the Filter text box, and this time search for AmazonS3FullAccess. Check the box next to this role.
9. Choose Next:Tags.
10. Choose Next:Review.
11. Enter VODLambdaRole for the Role name.
12. Choose Create role.
13. Type VODLambdaRole into the filter box on the Roles page and choose the role you just created.
14. On the Permissions tab, click on the Add Inline Policy link and choose the JSON tab.
15. Copy and paste the following JSON in the Policy Document Box. You will need to edit this policy in the next step to fill in the resources for your application.

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "\*",

"Effect": "Allow",

"Sid": "Logging"

},

{

"Action": [

"iam:PassRole"

],

"Resource": [

"arn:aws:iam::580151666824:role/MediaLiveCustomPolicy"

],

"Effect": "Allow",

"Sid": "PassRole"

},

{

"Action": [

"mediaconvert:\*"

],

"Resource": [

"\*"

],

"Effect": "Allow",

"Sid": "MediaConvertService"

}

]

}

1. Replace tag in the policy with the ARN for the vod-MediaConvertRole you created earlier.
2. Click on the Review Policy button.
3. Enter VODLambdaPolicy in the Policy Name box.
4. Click on the Create Policy button.

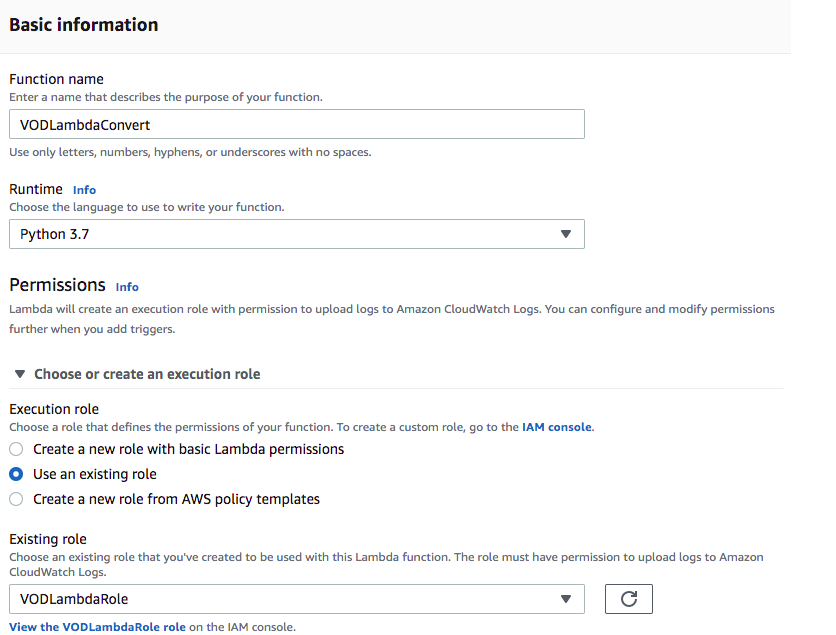
### 3. Create a Lambda Function for converting videos

AWS Lambda will run your code in response to events such as a putObject into S3 or an HTTP request. In this step you'll build the core function that will process videos using the MediaConvert python SDK. The lambda function will respond to putObject events in your S3 source bucket. Whenever a video file is added, the lambda will start a MediaConvert job.

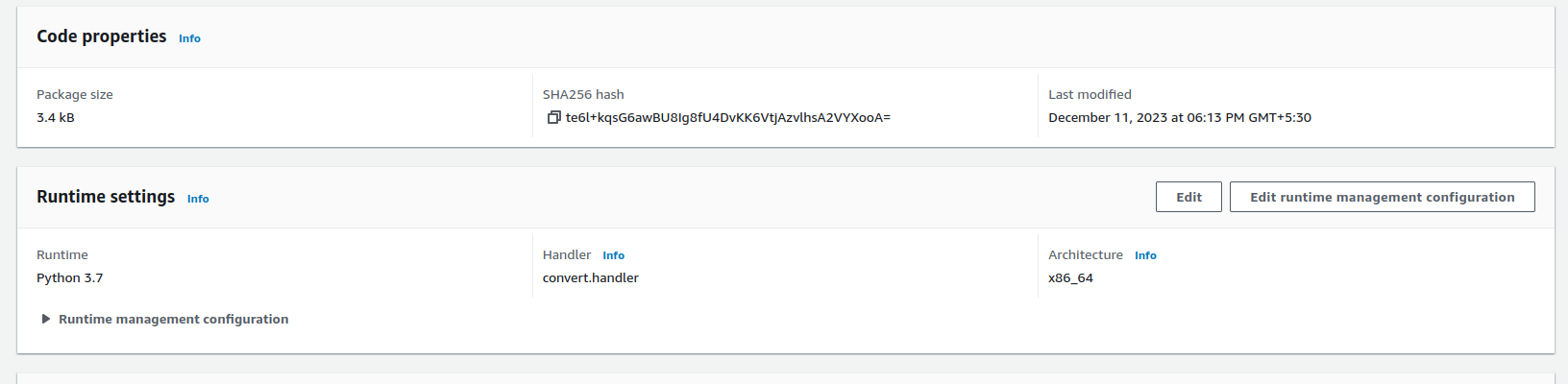
This lambda will submit a job to MediaConvert, but it won't wait for the job to complete. In a future module you'll use cloudwatch events to automatically monitor your MediaConvert jobs and take an action when they finish.

#### Step-by-step

1. Choose Services then select Lambda in the Compute section.
2. Choose Create function.
3. Choose the Author from scratch button.
4. On the Author from Scratch panel, enter VODLambdaConvert in the Function name field.
5. Select Python 3.7 for the Runtime.
6. Expand the Choose or create an execution role.
7. Select Use an existing role.
8. Select VODLambdaRole from the Existing Role dropdown.
9. Click on Create function.

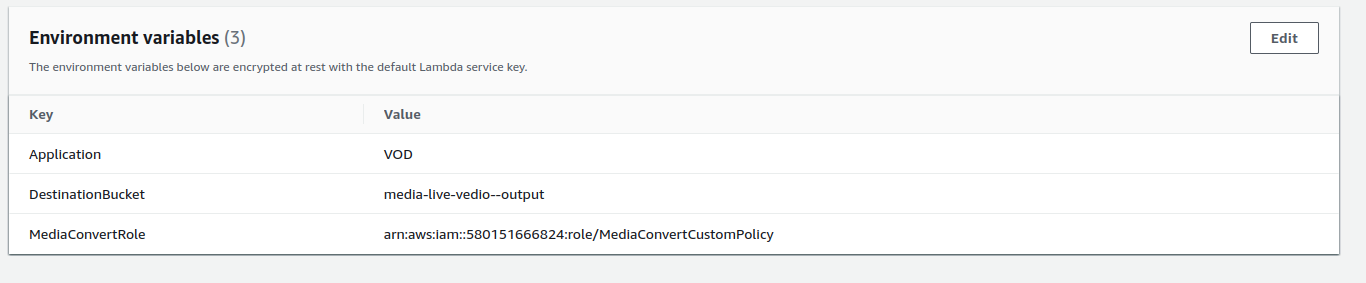


1. **On the Configuration tab of the VODLambdaConvert page, in the function code panel:**
2. Select Upload a file from Amazon S3 for the Code entry type
3. this zip file is simply the [convert.py script](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/convert.py) and the [job JSON](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/job.json) file provided in this git hub repository that you could zip up yourself.
4. Enter convert.handler for the handler field



1. On the Environment Variables panel of the VODLambdaConvert page, enter the following keys and values:

* DestinationBucket = vod-lastname (or whatever you named your bucket)
* MediaConvertRole = arn:aws:iam::ACCOUNT NUMBER:role/vod-MediaConvertRole
* Application = VOD



1. On the Basic Settings panel, set Timeout to 2 minutes.
2. Scroll back to the top of the page and click on the Save button.

### **Test the lambda**

From the main edit screen for your function, select Test.

Enter ConvertTest in the Event name box.

Copy and paste the following test event into the editor:

{

"Records": [

{

"eventVersion": "2.0",

"eventTime": "2017-08-08T00:19:56.995Z",

"requestParameters": {

"sourceIPAddress": "54.240.197.233"

},

"s3": {

"configurationId": "90bf2f16-1bdf-4de8-bc24-b4bb5cffd5b2",

"object": {

"eTag": "2fb17542d1a80a7cf3f7643da90cc6f4-18",

"key": "vodconsole/TRAILER.mp4",

"sequencer": "005989030743D59111",

"size": 143005084

},

"bucket": {

"ownerIdentity": {

"principalId": ""

},

"name": "rodeolabz-us-west-2",

"arn": "arn:aws:s3:::rodeolabz-us-west-2"

},

"s3SchemaVersion": "1.0"

},

"responseElements": {

"x-amz-id-2": "K5eJLBzGn/9NDdPu6u3c9NcwGKNklZyY5ArO9QmGa/t6VH2HfUHHhPuwz2zH1Lz4",

"x-amz-request-id": "E68D073BC46031E2"

},

"awsRegion": "us-west-2",

"eventName": "ObjectCreated:CompleteMultipartUpload",

"userIdentity": {

"principalId": ""

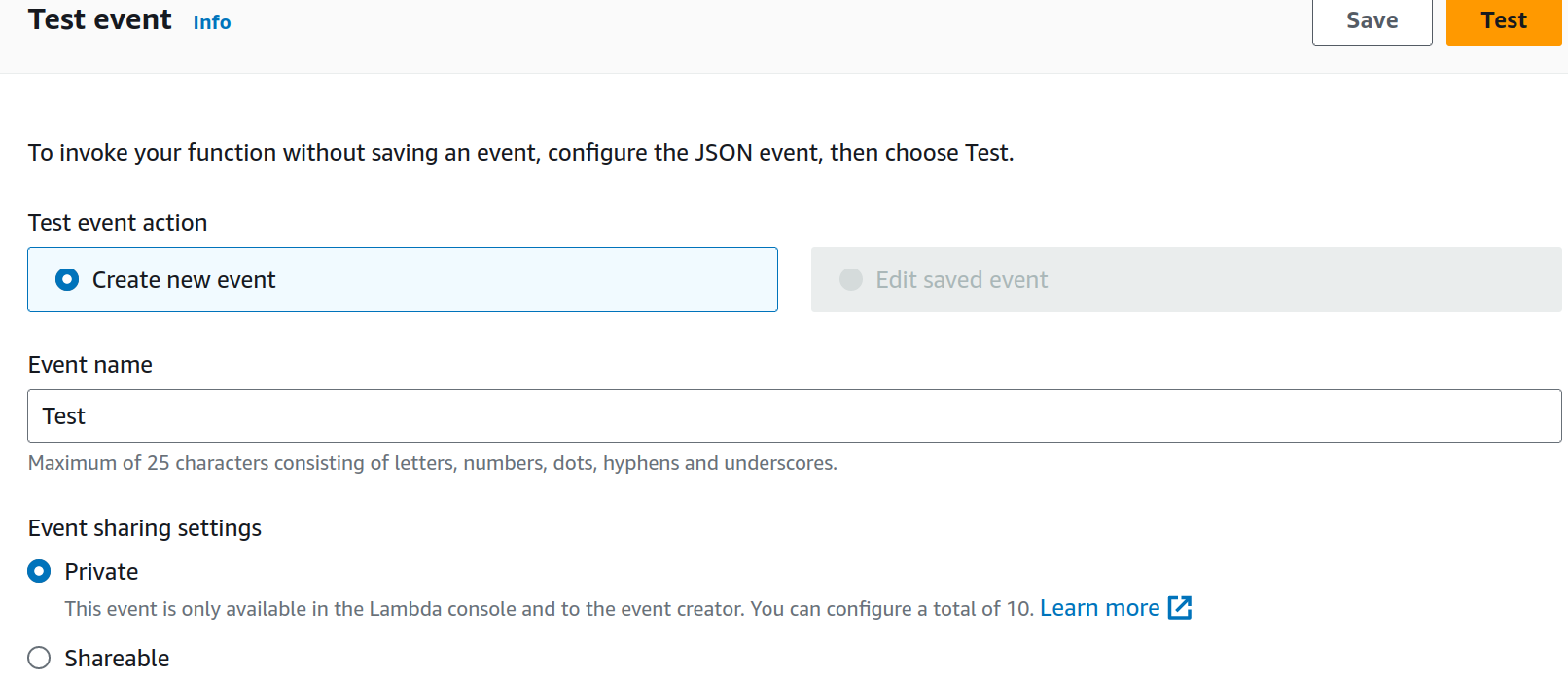
},

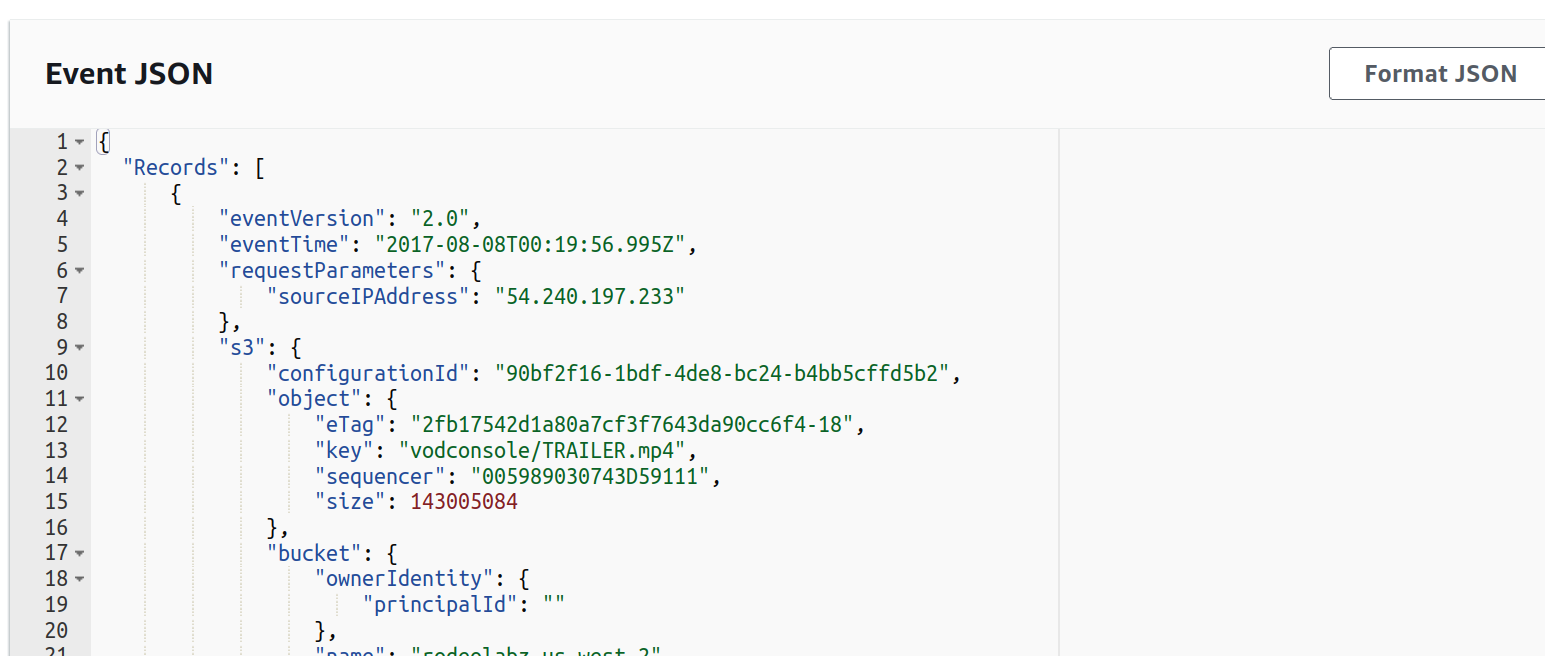
"eventSource": "aws:s3"

}

]

}





Click on Create button.

Then back on the main page, click on Test button.

Verify that the execution succeeded and that the function result looks like below;

{

"body": "{}",

"headers": {

"Access-Control-Allow-Origin": "\*",

"Content-Type": "application/json"

},

"statusCode": 200

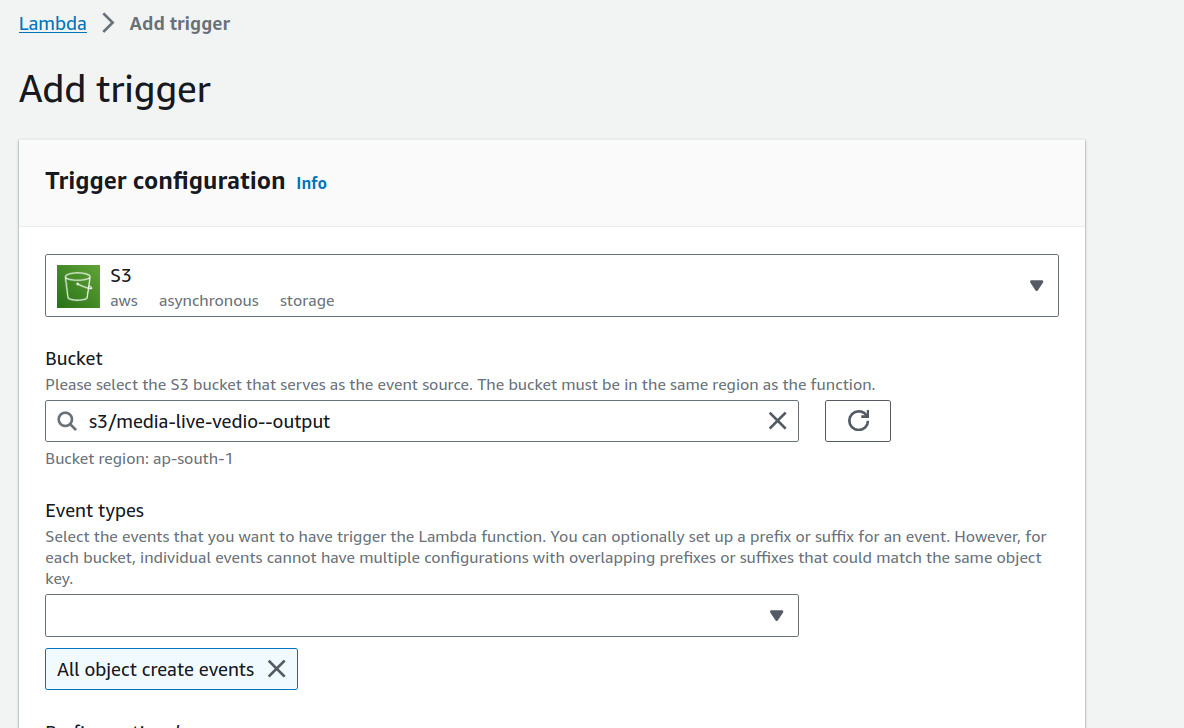
}

### **6. Create a S3 PutItem Event Trigger for your Convert lambda**

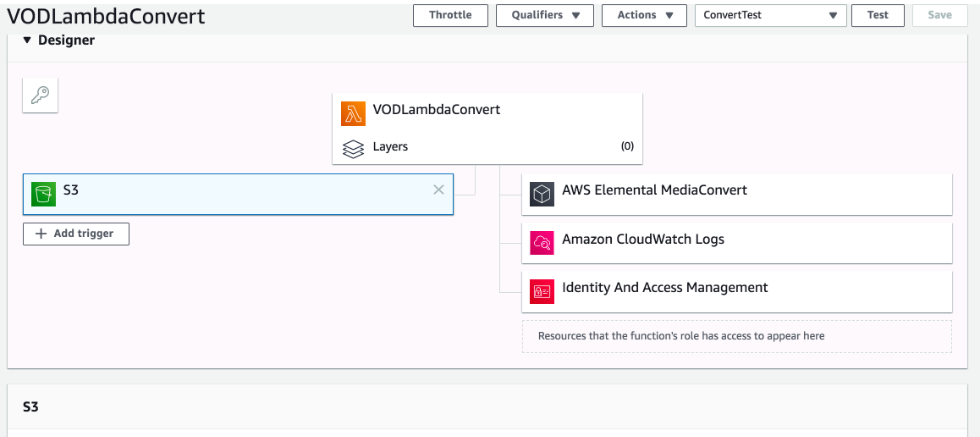
We built a lambda function that will convert a video in response to an S3 PutItem event. Now it's time to hook up the Lambda trigger to the watchfolder S3 bucket.

#### Step-by-step instructions

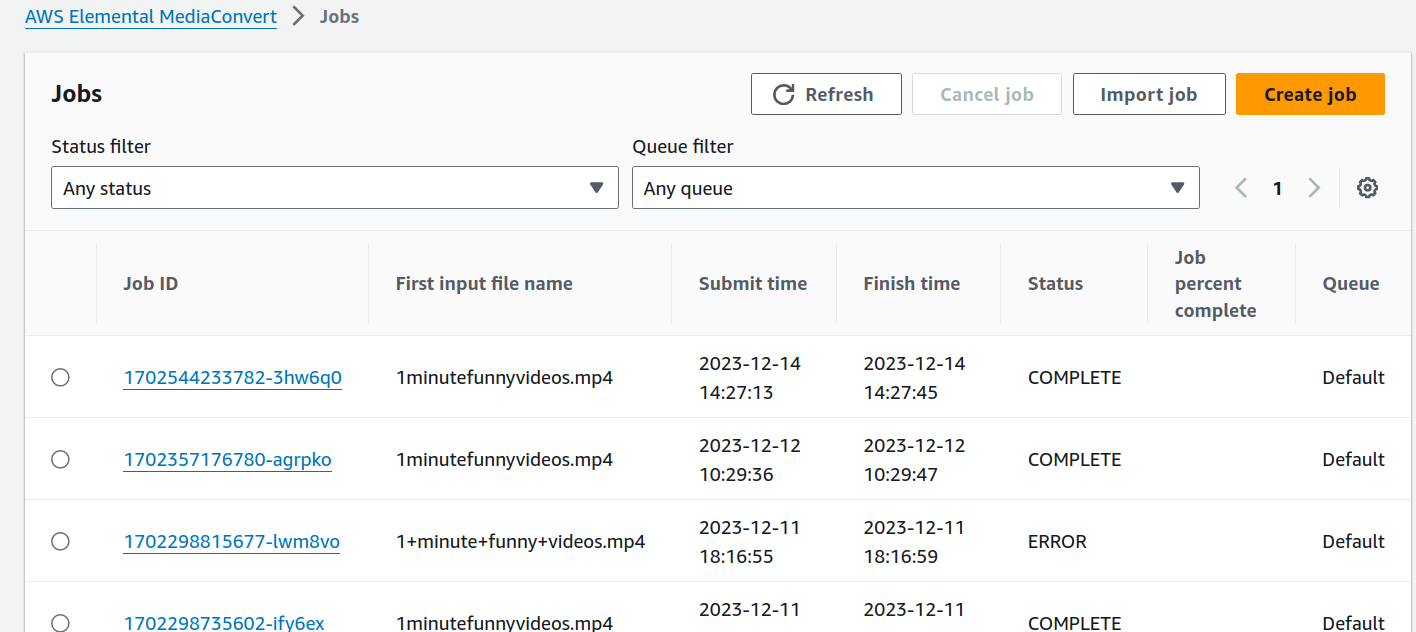
1. In the Configuration->Designer panel of the VODLambdaConvert function, click on Add trigger button.
2. Select S3 from the Trigger configuration dropdown.
3. Select vod-watchfolder-firstname-lastname or the name you used for the watchfolder bucket you created earlier in this module for the Bucket.
4. Select PUT for the Event type.

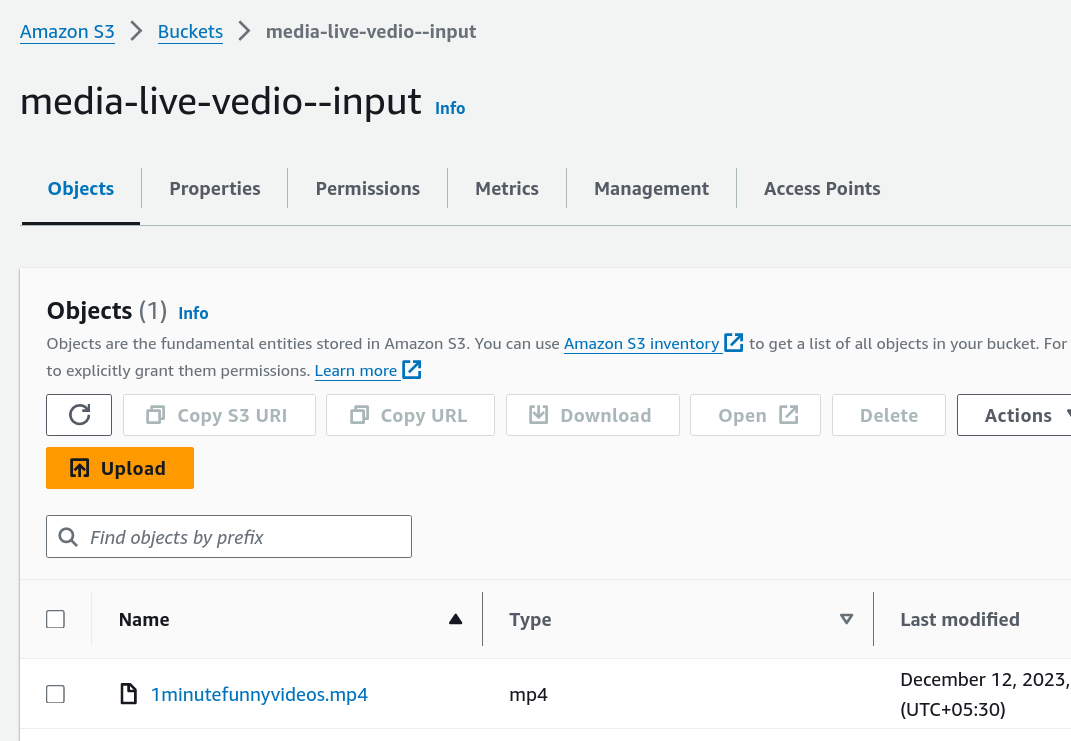


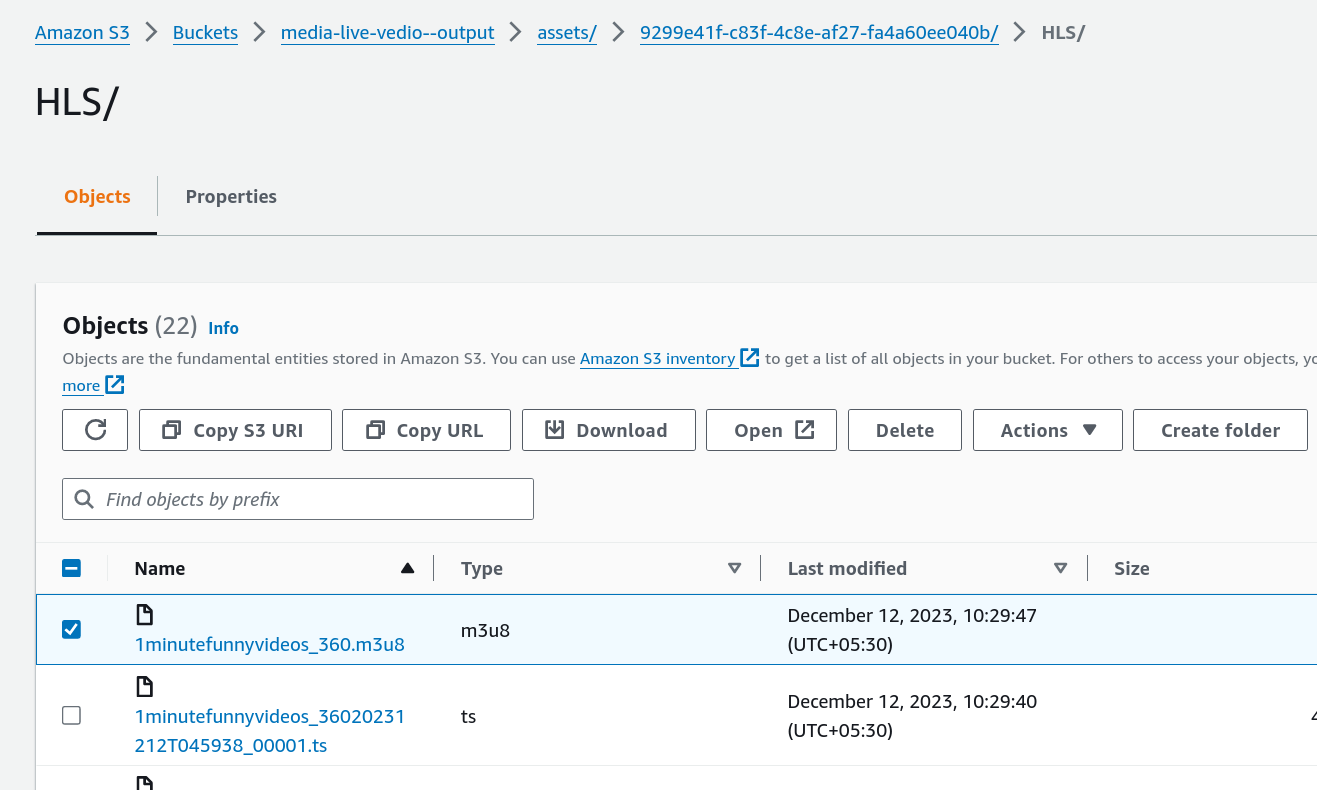
1. Leave the rest of the settings as the default and click the Add button.



* Open the S3 console overview page for the watchfolder S3 bucket you created earlier.
* Select Upload and then choose the file <vedio[.mp4>](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/test.mp4) from the directory for this lab module on your computer.
* Note the time that the upload completed.
* Open the MediaConvert jobs page and find a job for the input <vedio[.mp4>](https://github.com/aws-samples/aws-media-services-simple-vod-workflow/blob/master/7-MediaConvertJobLambda/test.mp4) that was started near the time your upload completed.
* Navigate to the MP4 or HLS output and play the test video by clicking on the test.mp4 output http resource.







Copy the object Url and play in HLS Streaming Tester or chrome browser