Converting Office Docs to PDF with AWS Lambda

ABSTRACT:

Now days every platform has reports and most of the time we need to convert the reports into PDF formats. I got the similar task few days back. I have done this conversion to PDF in past also but this time I decided to explore serverless approach.

There are multiple ways to implement this in AWS with servers and serverless: 1. EC2 Instances
2. ECS Faregate
3. EKS
4. Step Functions
5. Lambda

We are going to explore with Lambda. Lambda is an event-driven, serverless computing service which we can integrate with many other services like S3, SNS, DynamoDB etc.

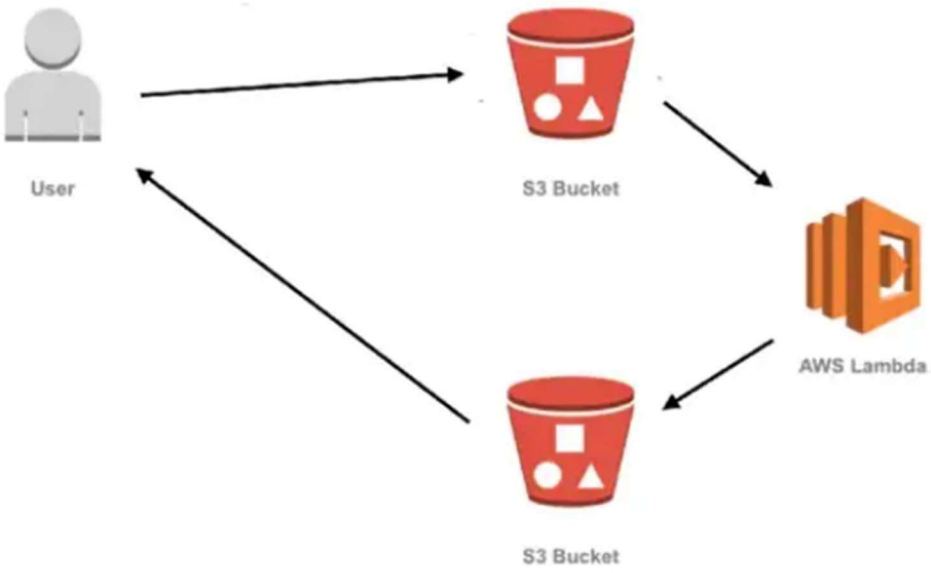
Why Serverless?

As we know conversion is a CPU intensive process. And converting documents to PDF at scale is a common problem. With Serverless approach we do not need to worry about scaling of our resources. Resources gets scaled automatically; scaling is handled by AWS. We are responsible only for our code, required memory and execution time.

Introduction:

To quickly and accurately convert the document which is in different formats to the desired pdf format. Many document converters, such as PDF to word converters and others, are available online. You must have experienced the need to convert an HTML page/document into PDF format. Similarly, it is often required to convert an excel sheet to a word document or other formats. AWS Lambda will allow you to develop an app that can rapidly convert documents from one format to the other. You can retrieve the required content and can format and convert the content to download or display on the webpage. You can deploy such an app in a job portal wherein the users often wish to convert their resume to another format.

Our plan…



1. User will upload Office Document to S3 bucket.
2. S3 bucket will trigger Lambda function with uploaded files details.
3. Lambda function will convert document with LibreOffice.
4. After conversion Lambda function will upload PDF to S3 Bucket.
5. After uploading you can write additional code to update details in Database or call any API to inform server.

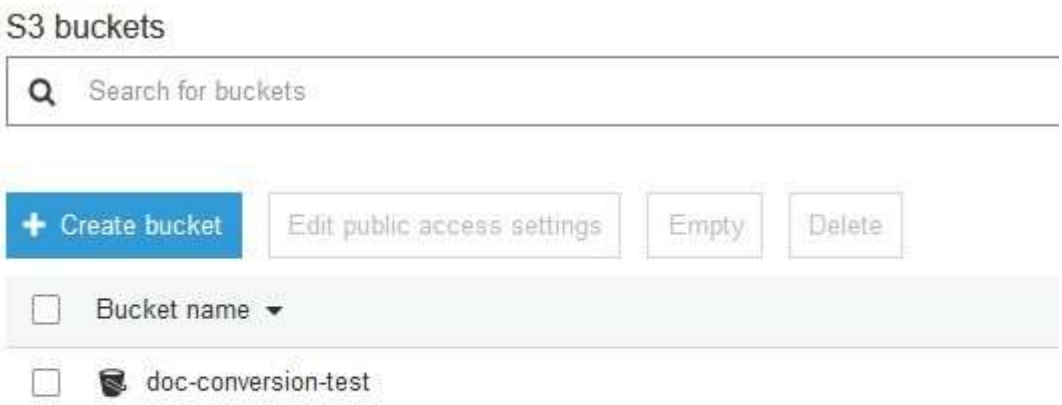
LIMITATION…

The only problem that we had in working with this project is, lambda has size limit on code package. You can upload 50mb zipped or 250 MB unzipped code. For document conversion we need to use LibreOffice which is 85 MB compress file and after extracting it becomes 300 MB. So, it will not fit in the limit.

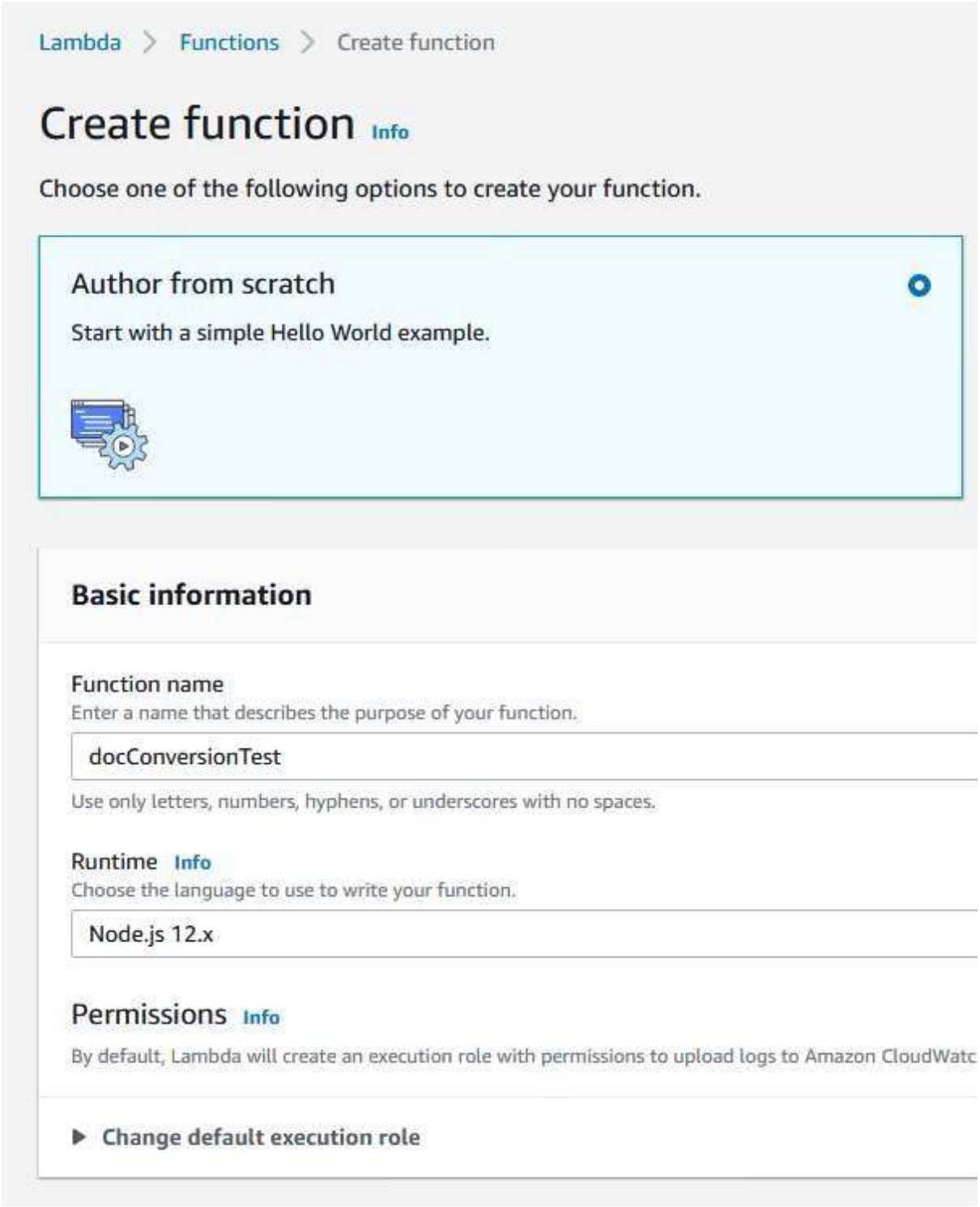
In Lambda we get 512 MB in `/tmp` location. With Lambda function we can pull in additional code and content in the form of layers. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. So, we will use layer for LibreOffice 85 MB compressed zip. LibreOffice is compressed with `brotli` , We will need to extract it `/tmp` to location before using it.

PROCEDURE: -

1. Create S3 Bucket ‘doc-conversion-test’

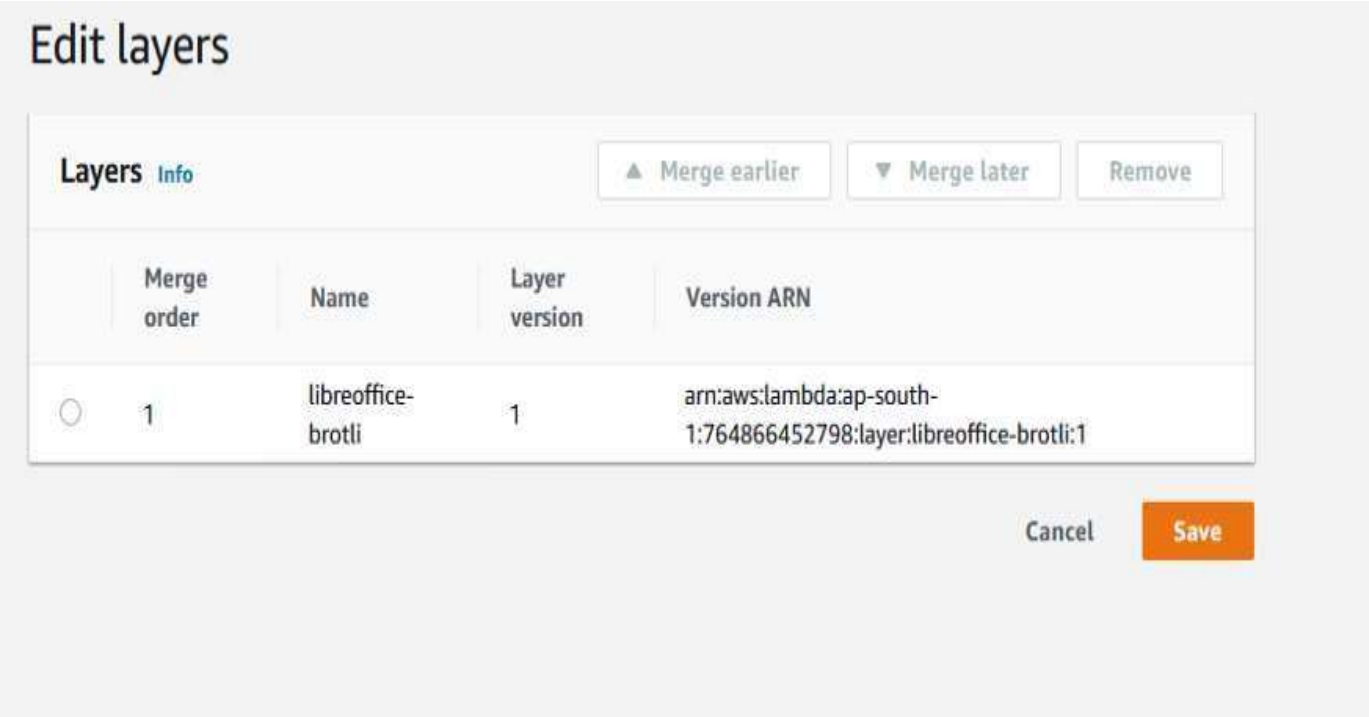


2. Create Lambda function



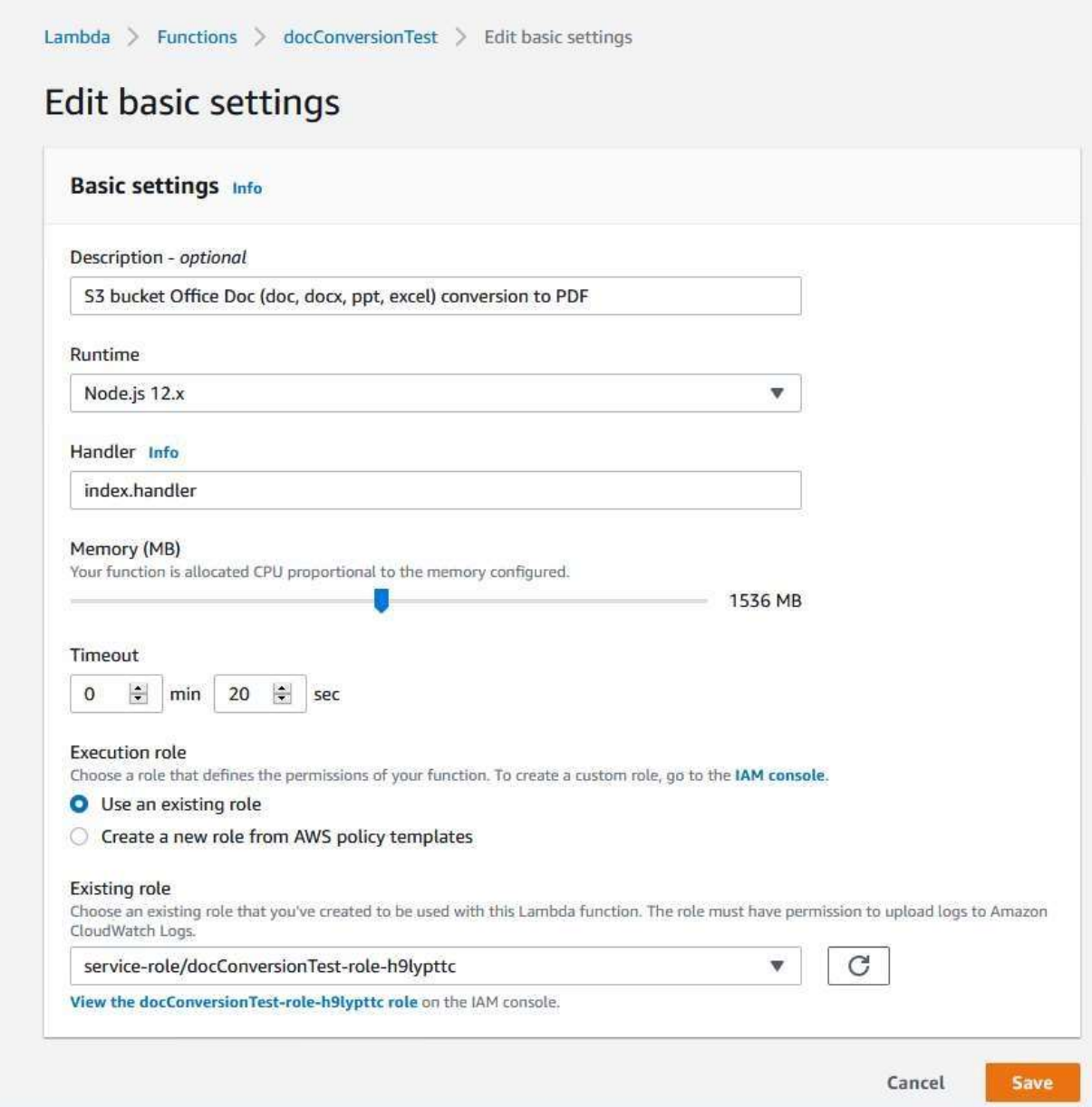
3. Add LibreOffice Layer to function

We can get started with LibreOffice, by clicking on layers and choose “Add a layer,” and “Provide a layer version ARN” and enter the obtained ARN or the following ARN: arn:aws:lambda:us-east-1:764866452798:layer:libreofficebrotli:1



Select ARN according to your region and add it.

4. Set Execution Timeout & Memory for our function:



You can decide on how much Memory and Timeout you need depending upon you average document size and complexity of document content. You can put some restrictions on size of the document for uploading. It will help you reduce Timeout and Memory allocation.

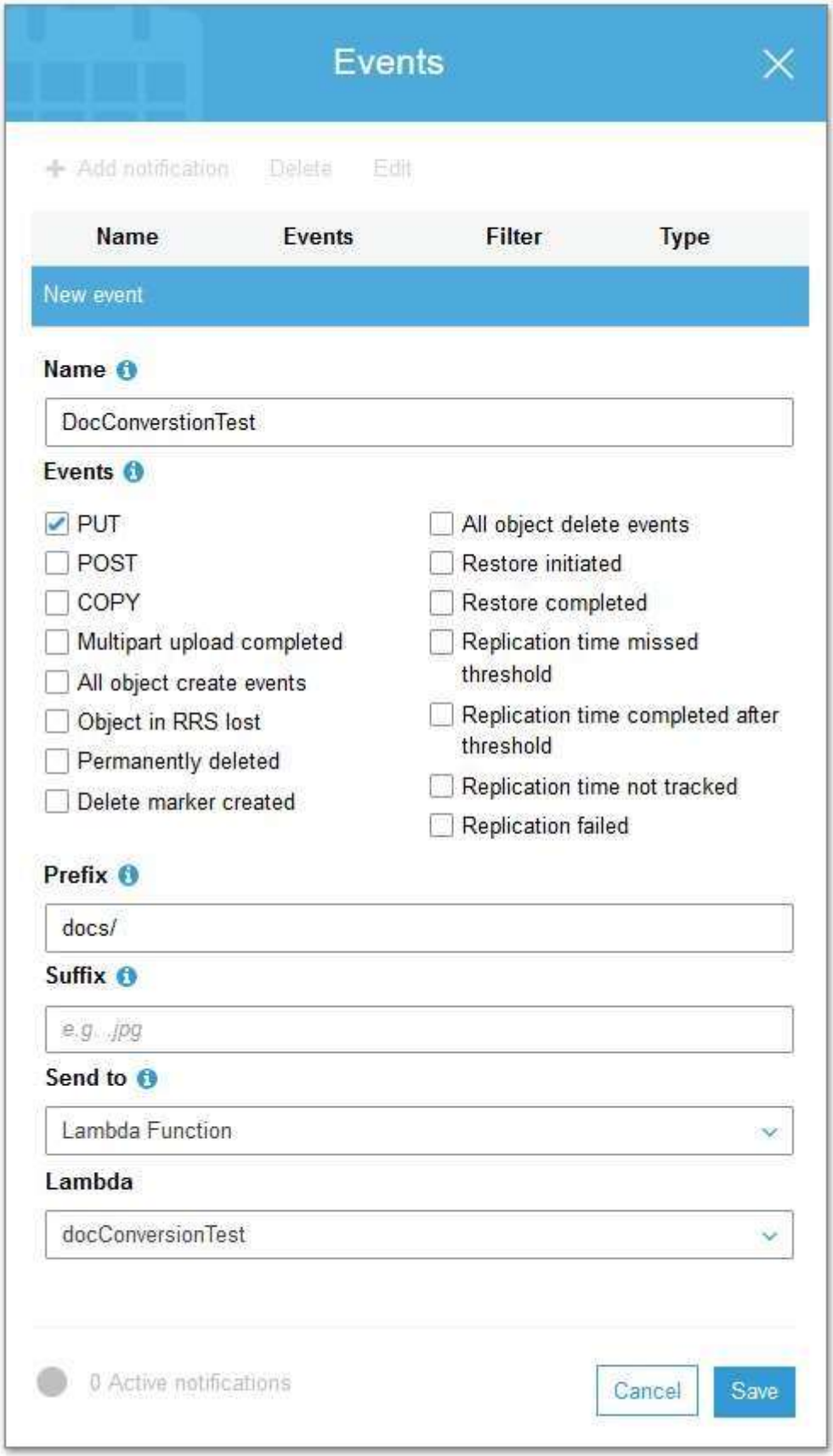
I have kept timeout to 20 secs because we are going to perform following tasks in our lambda function:

1. Extract LibreOffice to /tmp
2. Download Office Document from S3 bucket
3. Convert the document to PDF
4. Upload PDF to S3 Bucket

You can reduce or increase timeout depending upon your analysis. Lambda support 15 mins of Execution time. But the longer the execution time will cost you more. So, try to keep your code optimized to reduce execution time.

5. Setup S3 Bucket Trigger Event for Lambda Function

Go to your S3 bucket, under properties tab you will find Events.
Click on that and Add Notification.


Events
Add notification
Delete
Edit
Name
Events
Filter
Type
New event
Name
DocConverstionTest
Events
PUT
POST
COPY
Multipart upload completed
All object create events
Object in RRS lost
Permanently deleted
Delete marker created
All object delete events
Restore initiated
Restore completed
Replication time missed threshold
Replication time completed after threshold
Replication time not tracked
Replication failed
Prefix
docs/
Suffix
e.g. .jpg
Send to
Lambda Function
Lambda
docConversionTest
0 Active notifications
Cancel
Save

Add ‘Name’ for notification, Select PUT event so we will get notification once file upload is completed. Add ‘Prefix’ if you want or leave it blank. Select Lambda Function from ‘Send to’ dropdown box. Then select our Lambda function name and save it.

6. Let upload our code…

We are using Node.js 12.x as our runtime. We will install all required NPM packages on our local and then zip it for uploading code to Lambda.

```
const https = require('https');
const path = require('path');
const fs = require('fs');
var async = require('async');
const {writeFileSync} = require('fs');
const lambdafs = require('lambdafs');
const {execSync} = require('child_process');
var AWS = require('aws-sdk');

const inputPath = path.join( '/opt', 'lo.tar.br');
const outputPath = '/tmp/';
const bucketName = 'doc-conversion-test';

module.exports.handler = async (event, context) => {
  console.log(execSync('ls -alh /opt').toString('utf8'));

  try {
    // Decompressing
    let decompressed = {
      file: await lambdafs.inflate(inputPath)
    };

    console.log('output brotli de:----', decompressed);
  } catch (error) {
    console.log('Error brotli de:----', error);
  }

  try {
    console.log(execSync('ls -alh /opt').toString('utf8'));
  } catch (e) {
    console.log(e);
  }

  var body = "";
  //S3 put event
  body = event.Records[0].body;
  console.log('s3 bucket file name from event:', body);

  // get file from s3 bucket
  var s3fileName = body;
```

Here we are first extracting our LibreOffice to `/tmp` . After that we are taking S3 bucket filename from PUT event sent by S3 bucket to trigger the lambda function.

```
var newFileName = Date.now() + '.pdf';
var s3 = new AWS.S3({ apiVersion: '2006-03-01' });
var fileStream = fs.createWriteStream('/tmp/' + s3fileName);

var getObject = function (keyFile) {
    return new Promise(function (success, reject) {
        s3.getObject(
            { Bucket: bucketName, Key: keyFile },
            function (error, data) {
                if (error) {
                    reject(error);
                } else {
                    success(data);
                }
            }
        );
    });
}

let fileData = await getObject(s3fileName);
try {
    fs.writeFileSync('/tmp/' + s3fileName, fileData.Body);
} catch (err) {
    // An error occurred
    console.error('file write:', err);
}

const convertCommand = `export HOME=/tmp && /tmp/lo/instdir/program/soffice.bin
 --headless --norestore --invisible --nodefault --nofirststartwizard
  --nolockcheck --nologo --convert-to "pdf:writer_pdf_Export"
  --outdir /tmp /tmp/${s3fileName}`;
try {
    console.log(execSync(convertCommand).toString('utf8'));
} catch (e) {
    console.log(execSync(convertCommand).toString('utf8'));
}
console.log(execSync('ls -alh /tmp').toString('utf8'));

```

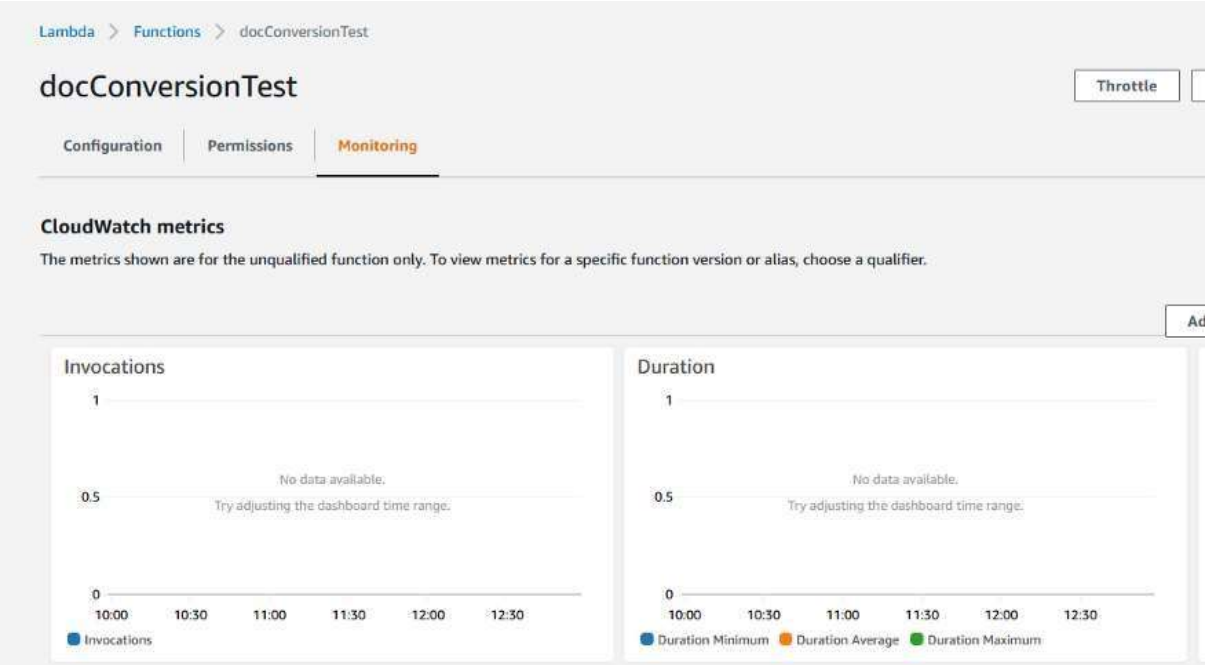
Now we are download the object to `/tmp` with AWS S3 library, and executing the LibreOffice document conversion command for PDF.

```
`export HOME=/tmp && /tmp/lo/instdir/program/soffice.bin -headless --norestore --invisible --nodefault -nofirststartwizard --nolockcheck --nologo --convert-to "pdf:writer_pdf_Export" --outdir /tmp /tmp/${s3fileName}`
```

```
78
79      function uploadFile(buffer, fileName) {
80          return new Promise((resolve, reject) => {
81              s3.putObject({
82                  Body: buffer,
83                  Key: fileName,
84                  Bucket: bucketName,
85              }, (error) => {
86                  if (error) {
87                      reject(error);
88                  } else {
89                      resolve(fileName);
90                  }
91              });
92          });
93      }
94
95
96      let fileParts = s3fileName.substr(0, s3fileName.lastIndexOf(".")) + ".pdf";
97      let fileB64data = fs.readFileSync('/tmp/' + fileParts);
98      await uploadFile(fileB64data, 'pdf/' + fileParts);
99      console.log('New file is converted to PDF and uploaded!!!');
100 };
```

Finally... we are uploading generated PDF back to S3 Bucket.

Now we are ready to test it. You can test it with sample event json in Lambda function console or just upload file to S3 Bucket. For debugging you can use Cloud Watch logs and X-Ray from monitoring tab.



You can explore more with Serverless framework or SAM for lambda.

And let me know if you find any other interesting solutions

:)

METHODS USED: -

1. We have used NODE.JS for our website.
2. Since it is serverless, the backend part is managed by AWS servers and it is stored in cloud.
3. Lambda is an event-driven, serverless computing service which we can integrate with many other services like S3, SNS, DynamoDB etc.
4. We have hosted website using S3 bucket from AWS service.

IMPLEMENTATION CODE: -

```
const https = require('https');

        const path = require('path');    const fs = require('fs');            var async = require('async');   const
{writeFileSync} = require('fs');          const lambdafs = require('lambdafs');     const {execSync} =
require('child_process');      var AWS = require('aws-sdk');


        const inputPath = path.join( '/opt', 'lo.tar.br');        const outputPath = '/tmp/';
                                                        const bucketName = 'doc-conversion-test';
                                module.exports.handler = async (event, context) => {           console.log(execSync('ls -alh
/opt').toString('utf8'));


                                                        try {

            // Decompressing                 let decompressed = {
                                                                file: await lambdafs.inflate(inputPath)
                                                        };


                                                                console.log('output brotli de:----', decompressed);
                                                        } catch (error) {
                                                                console.log('Error brotli de:----', error);
                                                }


                                                        try {
```

```
                                                    console.log(execSync('ls -alh
                                                  /opt').toString('utf8'));
        } catch (e) {           console.log(e);
                                                }


        var body = "";      //S3 put event
                                                  body = event.Records[0].body;
                                                      console.log('s3 bucket file name from event:', body);


        // get file from s3 bucket      var s3fileName = body;
                                                    var newFileName = Date.now()+'.pdf';
                                                var s3 = new AWS.S3({apiVersion: '2006-03-01'});

                                                  var fileStream =
                                                   fs.createWriteStream('/tmp/'+s3fileName);


        var getObject = function(keyFile) {       return new Promise(function(success, reject) {
                                                      s3.getObject(
            { Bucket: bucketName, Key: keyFile },          function (error, data) {
if(error) {                   reject(error);                } else {                        success(data);
                                                            }
                                                        }
                                                    );
                                                  });

                                                }


                                                     let fileData = await getObject(s3fileName);
                                                  try{
                                                  fs.writeFileSync('/tmp/'+s3fileName,
                                                fileData.Body);
                                                    } catch(err) {
                                                      // An error occurred
                                                          console.error('file write:', err);
                                                  }


                                                  const convertCommand = `export HOME=/tmp &&
                                                /tmp/lo/instdir/program/soffice.bin --headless -norestore --invisible --
                                                nodefault --nofirststartwizard -nolockcheck --nologo --convert-to
                                                "pdf:writer_pdf_Export"
                                                    --outdir /tmp /tmp/${s3fileName}`;
                                                    try {


                                                      console.log(execSync(convertCommand).toString('utf8'));
                                                    } catch (e) {


                                                      console.log(execSync(convertCommand).toString('utf8'));
                                                  }
                                                       console.log(execSync('ls -alh
                                                    /tmp').toString('utf8'));
```

```
function uploadFile(buffer, fileName) {
    return new Promise((resolve, reject) => {
        s3.putObject({
            Body: buffer,
            Key: fileName,
            Bucket: bucketName,
        }, (error) => {
            if (error) {
                reject(error);
            } else {
                resolve(fileName);
            }
        });
    });
}

let fileParts = s3fileName.substr(0,
    s3fileName.lastIndexOf(".")) + ".pdf";
    let fileB64data = fs.readFileSync('/tmp/'+fileParts);
    await uploadFile(fileB64data, 'pdf/'+fileParts);
    console.log('new pdf converted and uploaded!!!');
};
```

CONCLUSION: -

Hence, we have successfully completed the project. Our project provides a simple, convenient, reliable way of converting documents from any format to PDF format without making much complexity in the process.

REFERENCES:

1. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
2. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
3. https://www.mongodb.com/docs/drivers/node/current/quick-reference/