Objective

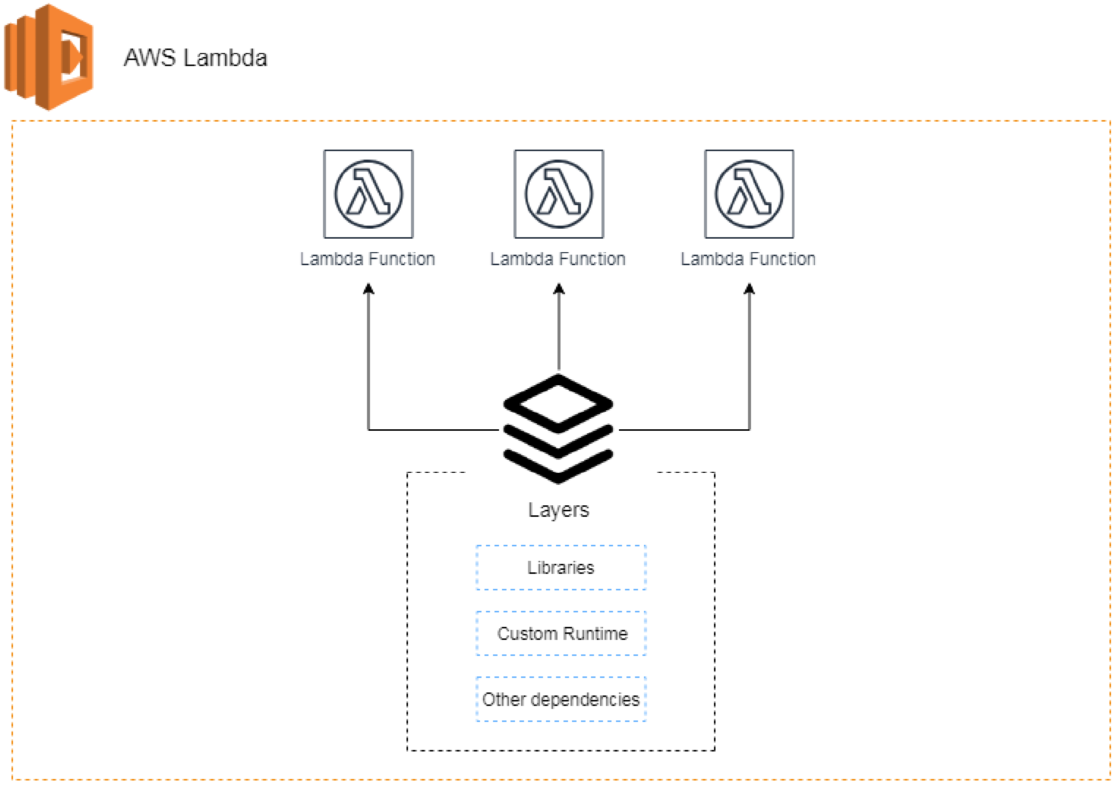
In this exercise we will explore two very exciting aspects of AWS Lambda - "Custom Layers" and "A lambda function in Python using layers".

The prelude

So let's get straight to it by understanding what exactly a layer is. In any system development we tend to assemble the common components in a single place with the intention of reuse. This thought process can manifest itself in several ways. For instance, in OOP we can think of base classes providing common functionality for sub classes. In case of an operating system we can think of the libraries being available to all applications.

The same thought process can be applied to Lambda as well. Imagine two lambda functions requiring the exact same dependency such as a library. This will mean that the two lambda functions will need to specify "requirements" and AWS will have to setup this particular library at the time of spinning up the individual lambda functions. This can cause setup and startup delays.

Layers address this exact inefficiency by having an environment that consists of the common library which can be provided to both lambda functions without the need for any installations. This will also mean that the individual lambda functions need not specify any requirements as well further simplifying the overall management of lambda code and environment.



The use case

Assume the following workflow -

1. User uploads a PDF document in a specific bucket
2. We use the bucket PUT event to trigger a lambda function
3. The lambda function will inspect the JSON and extract the bucket and filename
4. Read the file using the boto3 python library
5. Use the PyPDF library to parse the PDF and extract the metadata and content
6. Save the text content in a different bucket

We break the solution in two parts -

1. Create a common layer containing the 3rd party library dependency - PyPDF3
2. Write a lambda function in Python

Solution

Step 1

The solution outlined below might not work for AWS Academy account due to some service restrictions. We recommend to use your own account.

Use the following steps to create the custom layer -

1. Login to the AWS management console and select the region "N Virginia"
2. Create a T2 micro Ubuntu 20.04LTS instance. Open port 22 and create/reuse the PEM
3. Open a terminal window in your local laptop and SSH to the EC2 instance
4. Run the following commands in the terminal window -

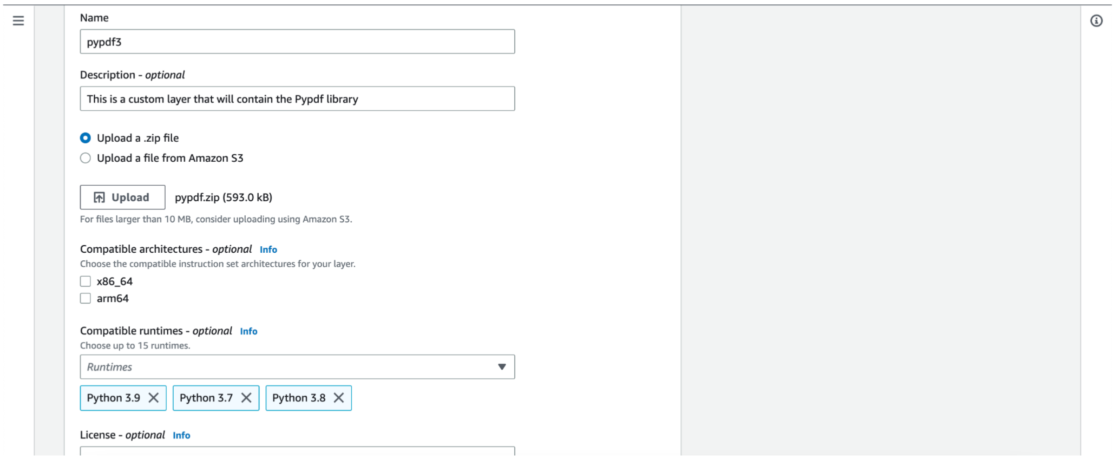
    a) sudo apt update
    b) sudo apt install zip -y
    c) sudo apt install python3-pip
    d) sudo chown ubuntu:ubuntu -R /opt
    e) cd /opt
    f) mkdir -p appbase/python
    g) sudo pip3 install pypdf3 -t appbase/python
    h) cd appbase/
    i) zip -r appbase.zip python

5. The zip file is now in the EC2 instance which needs to be downloaded to your local laptop. Use the below command to do so (please replace with your PEM file)

a) Open another terminal window and execute the following command

```
scp -i YOUR_PEM_FILE.pem
ubuntu@EC2_PUBLIC_IP:/opt/appbase/appbase.zip ./appbase.zip
```

6. Now go back to the AWS management console and visit the "Lambda" service
7. On the left navigation click on layers and create a new layer



Name
pypdf3

Description - *optional*
This is a custom layer that will contain the Pypdf library

● Upload a .zip file
○ Upload a file from Amazon S3

Upload pypdf.zip (593.0 kB)
For files larger than 10 MB, consider uploading using Amazon S3.

Compatible architectures - *optional* Info
Choose the compatible instruction set architectures for your layer.
□ x86_64
□ arm64

Compatible runtimes - *optional* Info
Choose up to 15 runtimes.
Runtimes
Python 3.9 ✕ Python 3.7 ✕ Python 3.8 ✕

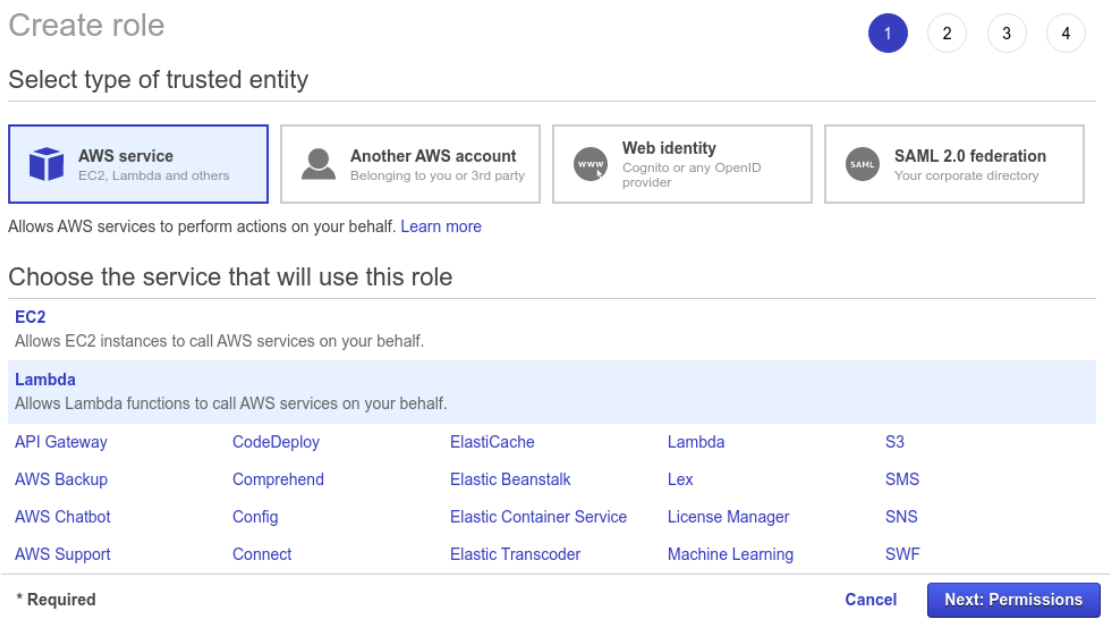
License - *optional* Info

8. Once the layer is created it will be listed in the set of layers that exist in your AWS account (notice that version 1 is automatically added)

Step 2

Now that the layer is ready to go, we need to create the Python lambda function. In order to ensure that the lambda function has access to all the other resources i.e. S3 and CloudWatch, ensure that you have an IAM role. Follow the below steps to create the role.

1. Visit the IAM role management console and click on “Create Role”. Chose “Lambda” as the service and click on “Next:Permissions” button



Create role

Select type of trusted entity

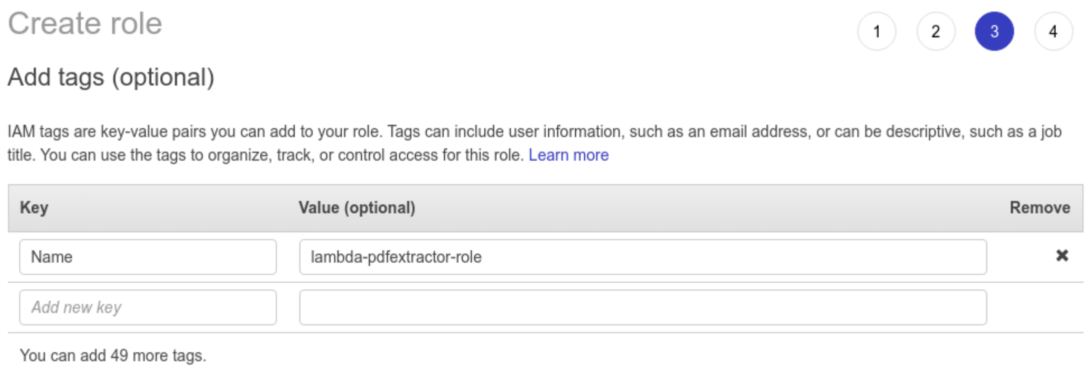
Allows AWS services to perform actions on your behalf. Learn more

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

2. Please search and add the following policies - “AmazonS3FullAccess” and “CloudWatchFullAccess”
3. Add the following tag to the role being created



Create role

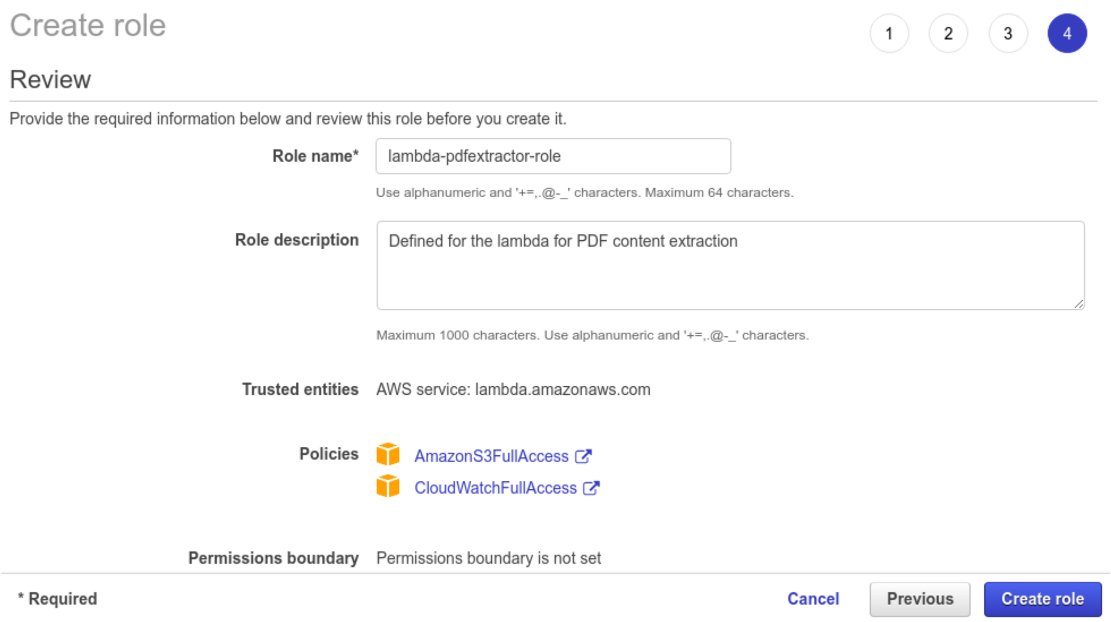
Add tags (optional)

IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. Learn more

Key	Value (optional)	Remove
Name	lambda-pdfextractor-role	✖
Add new key		

You can add 49 more tags.

4. In the final IAM role creation step, please fill in the form as follows and create the role by the name “lambda-pdfextractor-role”.



Create role

Review

Provide the required information below and review this role before you create it.

Role name* lambda-pdfextractor-role

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

Role description Defined for the lambda for PDF content extraction

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies AmazonS3FullAccess CloudWatchFullAccess

Permissions boundary Permissions boundary is not set

Step 3

Go to the S3 management console and create two buckets as follows

1. First bucket name: gl-src-[some random 5 digit number]
2. Second bucket: gl-dest-[some 5 digit random number]

Upload a PDF of under 100kb in the src bucket.

Step 4

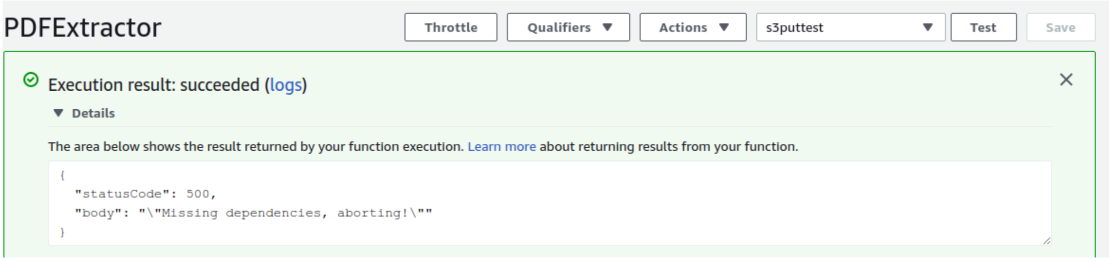
It is now time to create the lambda function using the layer and the role. Go back to the lambda management console and click on the "Functions" on the left navigation. Then click on "Create Function" button to start the function creation process

3. Select "*Author from scratch*" radio button from the top
4. Function name: *PDFExtractor*
5. Runtime: *Python 3.7*
6. Permissions: "*Use an existing role*" and select "*lambda-pdfextractor-role*" from the dropdown
7. Click on "*Create function*" button
8. In the lambda function code editor, remove the existing code and replace it with the code provided to you.
9. Environment variable key = TARGET_BUCKET
10. Environment variable value = gl-dest-[same random number]
11. Description: Extract the text content from the PDF file
12. Memory: 256MB

13. Timeout: 2min and 0secs
14. Click on “Save” button at the top of the page
15. Click on the drop down next to the “Test” button at the top of the page (next to the “Save” button)
16. Select “Configure test events” and a popup will open
17. In the Event Template dropdown select “Amazon S3 Put” option
18. Event name: s3puttest
19. In the JSON section, replace the JSON that is provided to you
20. Update the bucket name and object key to match your S3 src bucket and the PDF that you have uploaded to it in the JSON
21. Click on “Create” button to create the test case

Step 5

It is now time to test the lambda function by clicking on the “Test” button next to the dropdown showing the test cases. You will see that the execution is successful but it has gracefully exited because of missing dependency.



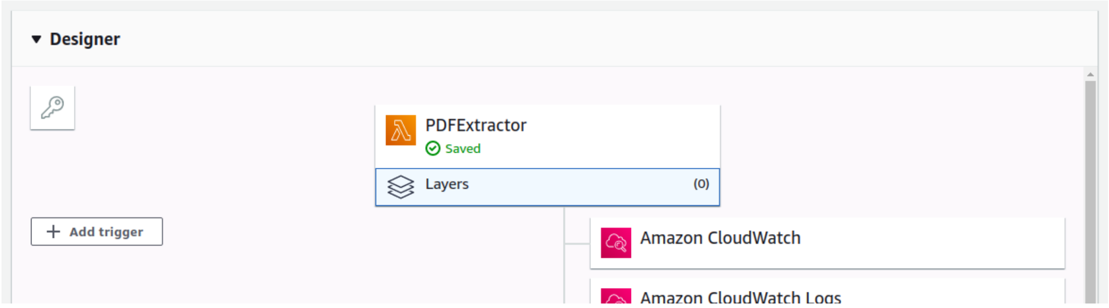
Execution result: succeeded (logs)

The area below shows the result returned by your function execution. Learn more about returning results from your function.

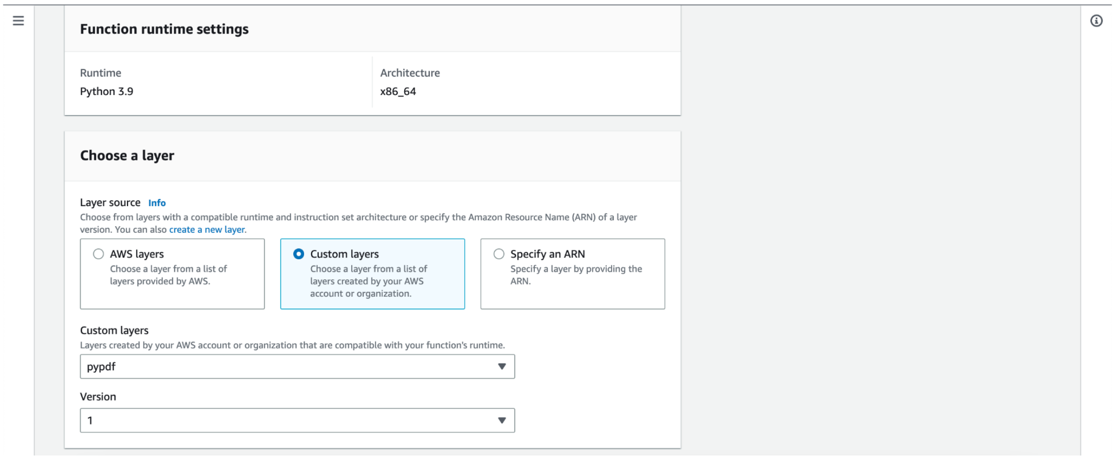
```
{
  "statusCode": 500,
  "body": "\"Missing dependencies, aborting!\""
}
```

The above is because the lambda function does not find the mandatory dependency of PyPDF. To fix the issue we will now associate the layer with the lambda function using the below steps -

1. Click on the “Layers” selection just below the lambda function name



2. Scroll down just a bit and you will see the screen now shows an empty list of attached layers
3. Click on the “Add a layer” button and select the layer created in the first step



Function runtime settings

Runtime
Python 3.9

Architecture
x86_64

Choose a layer

Layer source Info
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

○ AWS layers
Choose a layer from a list of layers provided by AWS.

● Custom layers
Choose a layer from a list of layers created by your AWS account or organization.

○ Specify an ARN
Specify a layer by providing the ARN.

Custom layers
Layers created by your AWS account or organization that are compatible with your function's runtime.
pypdf

Version
1

4. Click on the add button and you will see that this layer is now added to the lambda function. You will see "Layers (1)" just below the lambda function name in the "Designer" section.
5. Click on the "Save" button at the top of the screen
6. Check the destination bucket to see the text file which contains all the text content including any URLs.

Step 6

You can also connect this function from S3 source bucket via the events and test the integration by uploading a PDF in the bucket. **Finally, clean up all the resources created as part of this exercise** -

1. Lambda function
2. Layer
3. Source and destination buckets
4. The IAM role