

[Open in app](#)[Open in app](#)

towards
data science

[Follow](#)

603K Followers



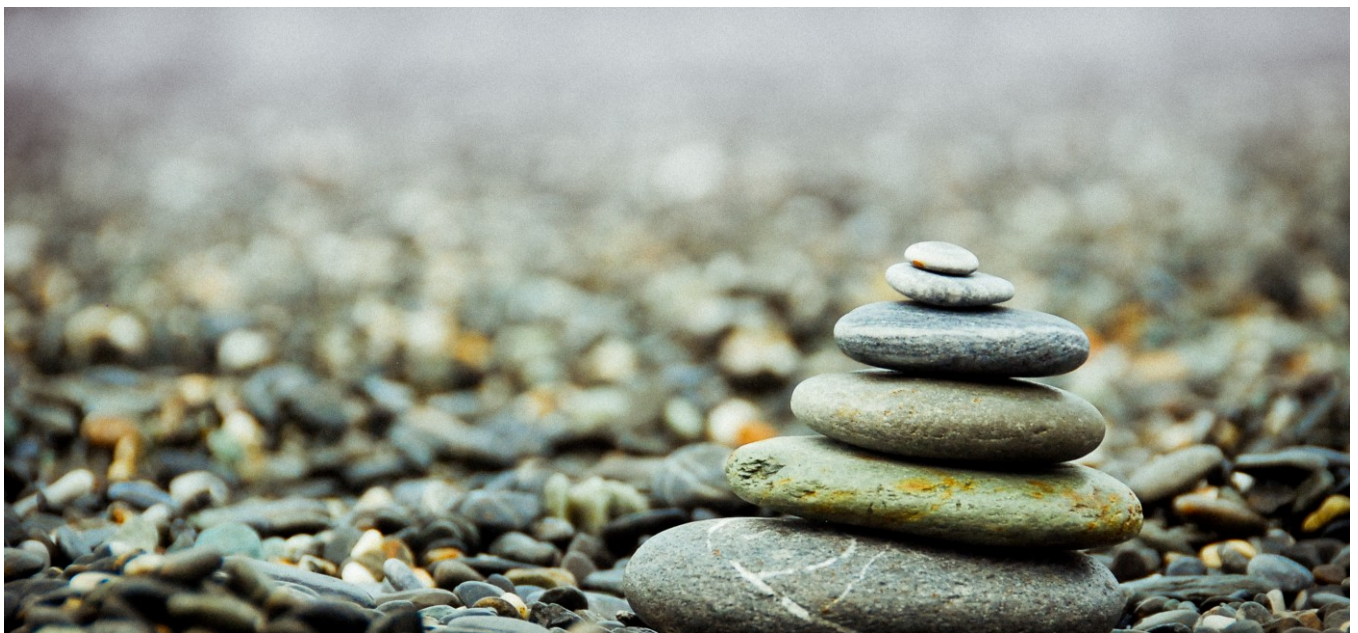
Building Custom Layers on AWS Lambda

How to build custom Python layers for your serverless application.



Israel Aminu Nov 13, 2020 · 4 min read

Many developers face issues when importing custom modules on AWS Lambda, you see errors like “No module named pandas” or “No module named numpy”, and most times, the easiest ways to solve this is to bundle your lambda function code with the module and deploy it on AWS lambda which at the end makes the whole project large and doesn't give room for flexibility. To solve this problem, we use something called **Layers** on AWS Lambda.



[Open in app](#)

Photo by [ZemkeHuang](#) on [Shutterstock](#)

What are Layers?

According to the docs, A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. With layers, you can use libraries in your function without needing to include them in your deployment package.

Layers let you install all the modules you need for your application to run, it provides that flexibility for you deploy your lambda function and even with layers you can even make your own custom code and add external functionality as a layer itself. It saves you the stress of managing packages and allows you to focus more on your code.

Benefits of Layers

- Makes your deployment package smaller and easily deployable
- The layer can also be used across other lambda functions.
- Make code changes quickly on the console.
- Lambda layers enable versioning, which allows you to add more packages and also use previous package versions when needed.

Now that we know what layers and see how useful it is, let's build one 🚀.

Step 1

Create a new directory and navigate to the directory on your computer:

```
israel@israel:~$ mkdir my-lambda-layer && cd my-lambda-layer
```

Step 2

Next, create a folder structure for the modules that you need to install:

```
israel@israel:~/my-lambda-layer$ mkdir -p aws-layer/python/lib/python3.7/site-packages
```

[Open in app](#)

different python version, change the *python3.7* in the folder above to the desired version.

Step 3

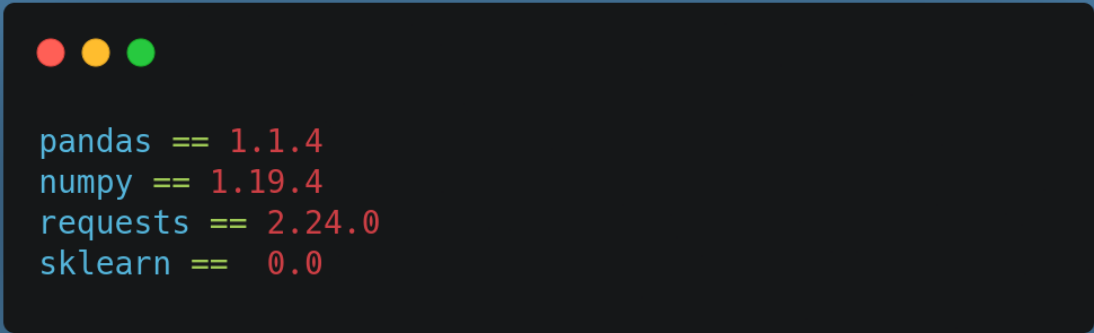
Let's install our libraries. To install just a single module for your application, use the following command, in this example I'll be using numpy.

```
israel@israel:~/my-lambda-layer$ pip3 install numpy --target aws-layer/python/lib/python3.7/site-packages
```

To install multiple modules, create a requirements.txt file in the base directory and add the modules with their respective versions:

```
israel@israel:~/my-lambda-layer$ nano requirements.txt
```

Add your modules, just like in the image below:



```
pandas == 1.1.4  
numpy == 1.19.4  
requests == 2.24.0  
sklearn == 0.0
```

Image by Author

Then install them with the command below:

```
israel@israel:~/my-lambda-layer$ pip3 install -r requirements.txt --target aws-layer/python/lib/python3.7/site-packages
```

[Open in app](#)

You should have a folder like this:

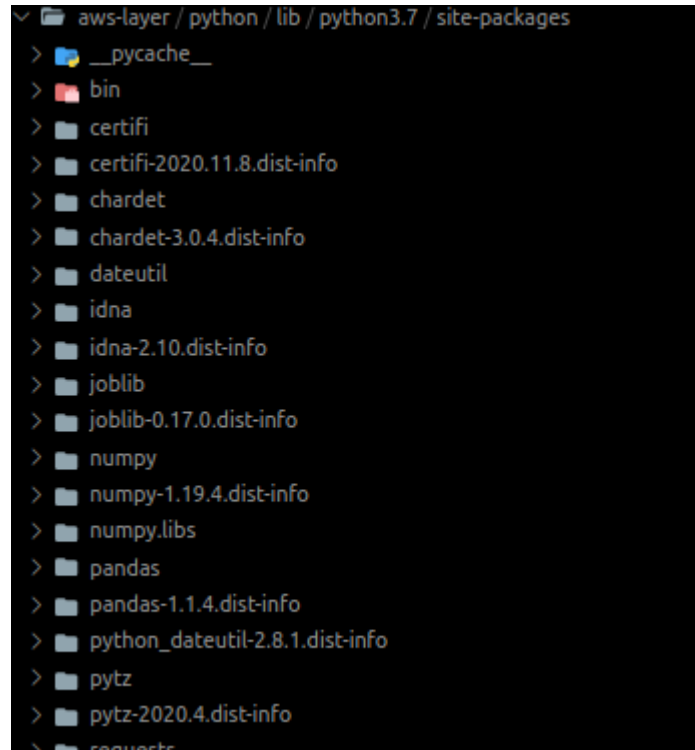


Image by author

Step 4

Next, we navigate to the *lambda-layer* directory and create a zip file for the layer that will be uploaded.

```
israel@israel:~/my-lambda-layer$ cd aws-layer
```

Now zip the entire folder:

```
israel@israel:~/my-lambda-layer/aws-layer$ zip -r9 lambda-layer.zip  
.
```

After zipping the packages it will have the name “lambda-layer.zip”

You can upload the zip file to your lambda layer using AWS CLI or using the AWS web Console, for this article I’ll be using the AWS CLI

Using the CLI

[Open in app](#)

```
aws lambda publish-layer-version \  
  --layer-name Data-Preprocessing \  
  --description "My Python layer" \  
  --zip-file fileb://lambda-layer.zip \  
  --compatible-runtimes python3.7
```

And you'll see the lambda layer created in your Web Console. Also, be sure to set your credentials and permissions on the AWS CLI to deploy the layer.

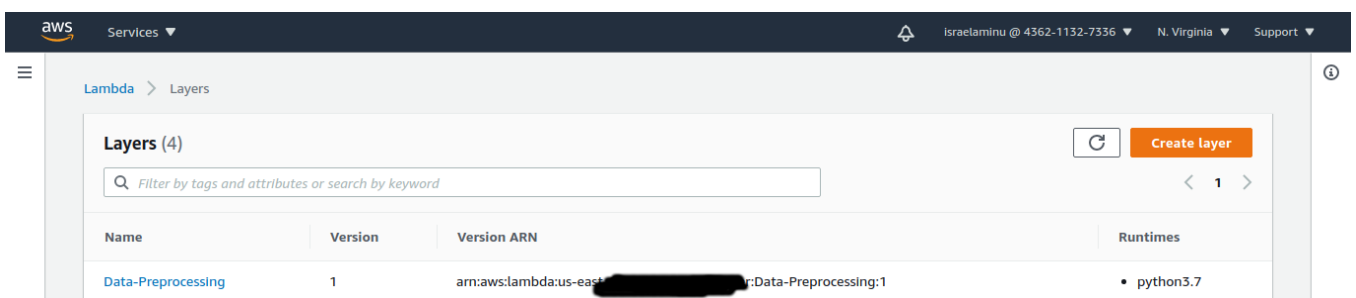


Image by author

Testing the Layer

To test the Lambda Layer, I simply created a new Lambda function and added my layer to it, you can do this by simply clicking on Layers, then click on the custom layer option and select the layer you just deployed to Lambda, at the end, you should see an image below:

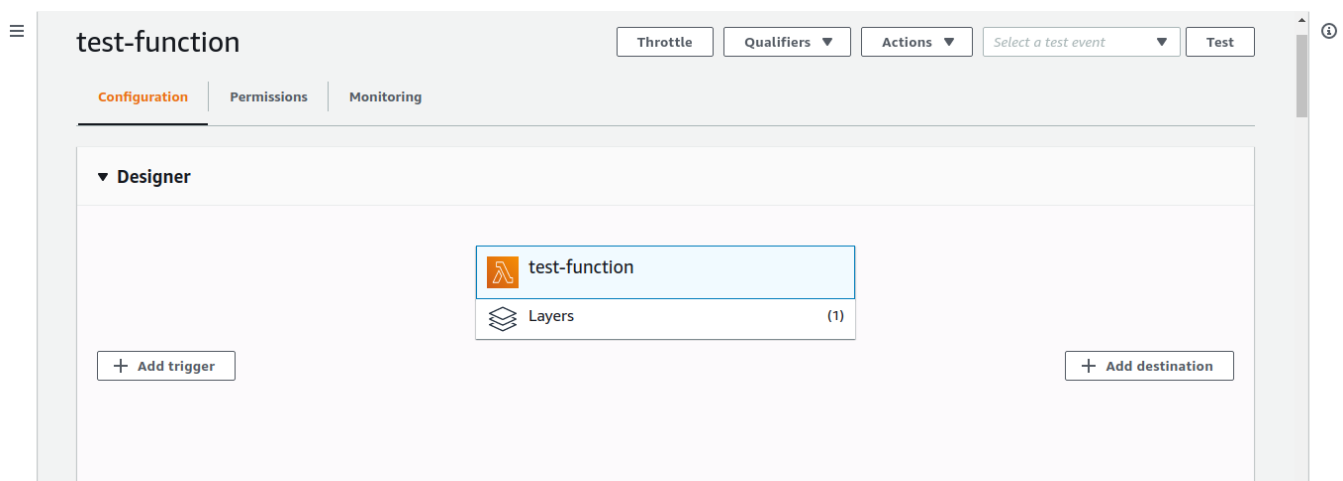


Image by author

Then on the function, I imported Pandas and numpy

[Open in app](#)

Image by author

Then I tested the function and got the following response

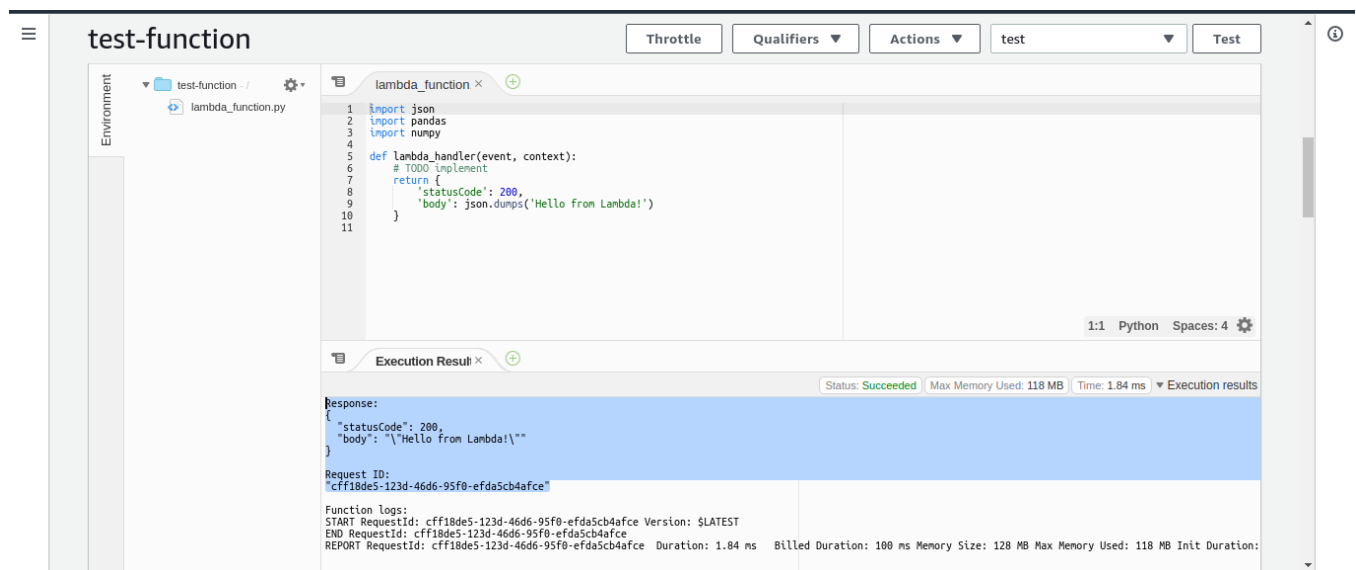


Image by author

From the image, our code was able to run successfully and we could import external packages for our application.

Summary

So far we've been able to see the beauty of layers and how powerful they are, you should always use them wherever you want to use external modules or even when you have your own custom code you've written.

Thanks for reading 😊.

Sign up for The Variable

By Towards Data Science

Open in app



Get this newsletter

Emails will be sent to business@bulrosa.com.
[Not you?](#)

AWS Lambda

Serverless

AWS

Python



About Write Help Legal

Get the Medium app

