ESRI
EXTERNAL

OptimizeRasters: AWS Lambda Implementation

User Documentation | November 15, 2018

380 New York Street
Redlands, California 92373-8100 USA
909 793 2853
info@esri.com
esri.com

esri | THE SCIENCE OF WHERE™

This document applies to OptimizeRasters Version 20180705 (see the OptimizeRasters.py header). OptimizeRasters and the associated GDAL library are currently provided as a prototype and for testing only. The functionality has not been exhaustively tested and is not currently covered under ArcGIS Support. Please address questions or suggestions related to this workflow to the OptimizeRasters GitHub forum or to ImageManagementWorkflows@esri.com.

Using OptimizeRasters with AWS Lambda

Esri's OptimizeRasters has the option to leverage AWS Lambda, which offers scalable, serverless, pay-as-you-go computing. The following document will address how to set up OptimizeRasters to delegate requests to an AWS Lambda function.

Requirements

- Amazon Web Services account
- CloudBerry Explorer (or a similar file manager for cloud storage)
- OptimizeRasters.py installed locally
- OptimizeRastersLambdaFX.zip file

Installation

Follow these steps to install OptimizeRasters as an AWS Lambda function.

Set up OptimizeRasters

1. Download and install the latest setup of OptimizeRasters (see documentation).
2. Download OptimizeRastersLambdaFX.zip. This's the OptimizeRasters AWS lambda function that needs to be installed using the AWS Console.
3. Use CloudBerry (or similar) to copy the OptimizeRastersLambdaFX.zip to an AWS S3 bucket.
4. Record the URL of the uploaded lambda_function.zip file, which will be used later by the AWS Console to point to the lambda function. To find the URL, select the uploaded zip file in the S3 bucket and copy the link address somewhere for later.

Note: The bucket housing the OptimizeRastersLambdaFX.zip must belong to the same AWS account where the OptimizeRasters lambda function will be set up. The OptimizeRastersLambdaFX.zip will not need any special permissions on S3 for the setup to work.

Set up an AWS service role

You'll need to create a service role that you will assign to the OR lambda function:

1. Log in to https://console.aws.amazon.com using your AWS credentials.
2. Using the AWS Console, navigate to Services > IAM > Policies.
3. Click **Create policy**.
4. Under the JSON tab, delete the default text, then copy and paste the following text (including all the curly brackets).

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
      "Effect": "Allow",
      "Action": [
       "logs:CreateLogGroup",
       "logs:CreateLogStream",
```

```
            "logs:PutLogEvents"
          ],
          "Resource": "arn:aws:logs:*:*:*"
        },
        {
          "Effect": "Allow",
          "Action": [
           "s3:GetObject",
           "s3:PutObject"
          ],
          "Resource": [
           "arn:aws:s3:::*"
          ]
        }
      ]
    }
```

5. Click **Review policy**.
6. Enter “OptimizeRasters” as the Name. Add a description for your policy. Click **Create policy**.
7. Using the AWS Console, select Services > IAM > Roles
8. Click **Create role** and select the following:
    a. Select type of trusted entity: AWS service
    b. Choose the service that will use this role: Lambda
9. Click **Next: Permissions**
10. Search for OptimizeRasters, then click the checkbox next to the OptimizeRasters policy you just created.
11. Click **Next: Review**
12. Enter “OptimizeRasters” as the Name. Click **Create role**.
13. Click **Create role**.

Set up the OptimizeRasters Lambda function

The following steps will walk through how to set up the OptimizeRasters lambda function in AWS.

1. Log in to https://console.aws.amazon.com using your AWS credentials.

    Note: The bucket housing the lambda_function.zip file and your lambda function must be located in the same AWS region. Confirm this is the case before continuing.
2. Using the AWS Console, select Services > Lambda.
3. Click **Create function**.
4. Under the “Author from scratch” option, enter the following:
    a. Name: OptimizeRasters

   b. Runtime: Python 2.7
   c. Role: Choose an existing role
   d. Existing role: OptimizeRasters

5. Click **Create function**.
6. Scroll down to the “Function code” section and choose the following options:
   a. Code entry type: Upload a file from Amazon S3
   b. Amazon S3 link URL: URL of the lambda_function.zip uploaded to S3
7. Scroll down to the “Environmental Variables” section and enter the following:
   a. Key: LD_LIBRARY_PATH
   b. Value: /var/task/GDAL/bin
8. Scroll down to the “Basic settings” section and set the following parameters:
   a. Description: OptimizeRasters
   b. Memory: 640 MB
   c. Timeout 5 min 0 sec
9. Leave all other defaults. Click **Save** at the top right of the screen.
10. Navigate to Services > Lambda and verify that the newly added function is available.
11. Select Services > Simple Notification Service.
12. Click **Create topic**. Set the following parameters:
   a. Topic name: ORLambda
   b. Display name: ORLambda
13. Click **Create topic**.
14. Click **Create subscription**. Set the following parameters (leave other defaults):
   a. Protocol: AWS Lambda
   b. Endpoint: Select your newly added OptimizeRasters function
15. Click **Create subscription**.
16. Record the Topic ARN at the top of the page. (It will be similar to: arn:aws:sns:us-east-1:756397045908:ORLambda)

The configuration steps on the AWS Console are now complete.

Client-side set up

You will need to connect the OptimizeRasters lambda function with OptimizeRasters on Windows. To do this, edit the c:\Users\%username%\.aws\credentials file and enter the following credential entry:

```
[aws_lambda]
aws_access_key_id =
<your_aws_access_key_where_the_lambda_function_was_setup>
aws_secret_access_key =
<your_aws_secret_key_where_the_lambda_function_was_setup>
sns_arn = <arn_entry_from_step_16 (e.g. arn:aws:sns:us-east-1:936065890666:ORLambda)>
```

Save the file and exit the editor.

Using the OptimizeRasters lambda function

Once the above steps have been completed successfully, you'll be able to issue OptimizeRasters requests to delegate processing using the AWS OptimizeRasters Lambda function instead of the local machine. The remote OptimizeRasters lambda function can be invoked as an asynchronous or synchronous operation, depending on the command line flag used:

a. **-op=lambda**
   This mode uses the AWS SNS channel shown earlier. In this asynchronous mode, the status of the remote processing is not returned, and the local job file created will remain at the OptimizeRasters root folder.

a. **-op=lambda:function:OptimizeRasters**
   This method calls the remote OptimizeRasters function synchronously. The remote function will return a response to update the active OptimizeRasters job file. If all the rasters have been processed successfully, the local job file will be moved into the Job folder.

Following is the standard command line format for calling the OptimizeRasters lambda function. The usage shows calling the lambda OptimizeRasters asynchronously by using the **-op=lambda** flag:

python.exe C:\Image_Mgmt_Workflows\OptimizeRasters\OptimizeRasters.py -clouddownload=true -clouddownloadtype=Amazon -inputbucket=<input_bucket_name> -input=<S3_input_folder_path> -inputprofile=<input_profile_in_credential_file> -tempinput=<local_path_for_temp_input_directory> -cloduploa d=true -clouduploadtype=Amazon -outputbucket=<output_bucket_name> -output=<S3_output_folder_path> -outputprofile=<output_profile_in_credential_file> -tempoutput=<local_path_for_temp_output_directory> -config=c:\Image_Mgmt_Workflows\OptimizeRasters\Templates\Imagery_to_MRF_LERC.xml **-op=lambda**

The **-queuelength** parameter can be used to control how many rasters/files to feed into a single OptimizeRaster lambda function; the parameter takes an integer value (1 to x).

Note: The -queuelength flag should be used only when you're sure the S3 source path folder contains rasters that can be processed individually, without any associated/required metadata files (TIL files, for example). Otherwise, you risk splitting up a multi-file raster across multiple lambda functions.

Verifying the OptimizeRasters lambda function

If OptimizeRasters.py successfully delegates the local request to the remote OptimizeRasters lambda function using the -op=lambda command line flag, you will see text similar to what's shown below:

log-msg: [Resume] Creating job (OR_20161027T182437278000.orjob)
log-msg: **Using AWS Lambda**..
log-msg: Invoke using (SNS)
log-msg: Lambda working on the (RequestID) [ba99b6bd-f07f-5bdb-a401-e631e01634a5]
log-msg: Lambda working on the (RequestID) [58c50795-e4e4-596a-9b8c-547a65fdde14]
…
log-status: [OK]
Time taken> 7.78s