

# HOW TO DEFINE AND RUN A JOB IN AWS GLUE



Here we show how to run a simple job in Amazon Glue.

The basic procedure, which we'll walk you through, is to:

- Create a Python script file (or PySpark)
- Copy it to Amazon S3
- Give the Amazon Glue user access to that S3 bucket
- Run the job in AWS Glue
- Inspect the logs in Amazon CloudWatch

## Create Python script

First we create a simple Python script:

```
arr=  
  
for i in range(len(arr)):  
    print(arr)
```

## Copy to S3

Then use the Amazon CLI to create an S3 bucket and copy the script to that folder.

```
aws s3 mb s3://movieswalker/jobs
aws s3 cp counter.py s3://movieswalker/jobs
```


## Configure and run job in AWS Glue

Log into the Amazon Glue console. Go to the Jobs tab and add a job. Give it a name and then pick an Amazon Glue role. The role **AWSGlueServiceRole-S3IAMRole** should already be there. If it is not, add it in IAM and attach it to the user ID you have logged in with. See instructions at the end of this article with regards to the role.

### Configure the job properties


**Name**

**IAM role** ⓘ


AWSGlueServiceRole-S3IAMRole 

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role.](#)

**Type**

Python shell 

**Python version**


Python 3 (Glue Version 1.0) 

**This job runs**

An existing script that you provide

A new script to be authored by you

**S3 path where the script is stored**

s3://movieswalker/jobs 

The script editor in

Amazon Glue lets you change the Python code.

The job run failed. For help, post the error message on the [AWS Glue forum](#) or contact [AWS support](#) with the job run ID: jr\_93c9ac1053c3a1c1907f5a3887cd754e5c1c9c722bd51ae56bb55754db07b99d

Job: PrintArray

Action ▾

Save

Run job

```
1
2 arr=[1,2,3,4,5]
3
4 for i in range(len(arr)):
5     print(arr[i])
6
7
8
```

Logs

Schema

Aug 16, 2020, 12:45:44 PM Pending execution

This screen shows that you can pass run-time parameters to the job:

## Parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

- ▶ Tags
- ▶ Security configuration, script libraries, and job parameters

Only job **PrintArray** is run. Jobs dependent on the completion of job **PrintArray** will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

Run job

Run the job. When you run it, if there is any error you are directed to CloudWatch where you can see that. The error below is an S3 permissions error:

CloudWatch > CloudWatch Logs > Log groups > /aws-glue/python-jobs/error > [Switch to the original interface.](#)  
jr\_93c9ac1053c3a1c1907f5a3887cd754e5c1c9c722bd51ae56bb55754db07b99d

**Log events** Actions Create Metric Filter

Filter events  Clear 1m 30m 1h 12h Custom

Timestamp	Message
	There are older events to load. <a href="#">Load more.</a>
2020-08-16T12:45:58.567+03:00	<pre> Traceback (most recent call last):   File "/tmp/runscript.py", line 114, in &lt;module&gt;     temp_file_path = download_user_script(args.scriptLocation)   File "/tmp/runscript.py", line 91, in download_user_script     download_from_s3(args.scriptLocation, temp_file_path)   File "/tmp/runscript.py", line 81, in download_from_s3     s3.download_file(bucket_name, s3_key, new_file_path)   File "/usr/local/lib/python3.6/site-packages/boto3/s3/inject.py", line 172, in download_file     extra_args=ExtraArgs, callback=Callback)   File "/usr/local/lib/python3.6/site-packages/boto3/s3/transfer.py", line 307, in download_file     future.result()   File "/usr/local/lib/python3.6/site-packages/s3transfer/futures.py", line 106, in result     return self._coordinator.result()   File "/usr/local/lib/python3.6/site-packages/s3transfer/futures.py", line 265, in result     raise self._exception   File "/usr/local/lib/python3.6/site-packages/s3transfer/tasks.py", line 255, in _main     self._submit(transfer_future=transfer_future, **kwargs)   File "/usr/local/lib/python3.6/site-packages/s3transfer/download.py", line 345, in _submit     **transfer_future.meta.call_args.extra_args   File "/usr/local/lib/python3.6/site-packages/botocore/client.py", line 357, in _api_call     return self._make_api_call(operation_name, kwargs) </pre>

Here is the job run

history.

[Add job](#) Action

**History** **Details** **Script**

[View run metrics](#) [Rewind job bookmark](#)

Run ID	Retry	Run attempt	Run status	Error	Output	Logs	Error logs	Glue version	Maximal capacity	Triggered by	Start time	End time
<a href="#">jr_029c64f3...</a>	-	Succeeded				<a href="#">Logs</a>		1.0	0.0...		17 ...	17 ...
<a href="#">jr_14b0d642...</a>	-	Failed				<a href="#">Logs</a>	<a href="#">Error logs</a>	1.0	0.0...		17 ...	17 ...
<a href="#">jr_a8fc8290...</a>	-	Failed				<a href="#">Logs</a>	<a href="#">Error logs</a>	1.0	0.0...		17 ...	17 ...
<a href="#">jr_8eb1daa1...</a>	-	Failed				<a href="#">Logs</a>	<a href="#">Error logs</a>	1.0	0.0...		16 ...	16 ...
<a href="#">jr_5a675014...</a>	-	Failed				<a href="#">Logs</a>	<a href="#">Error logs</a>	1.0	0.0...		16 ...	16 ...

Here is the log showing that the Python code ran successfully. In this simple example it just printed out the numbers 1,2,3,4,5. Click the **Logs** link to see this log.

CloudWatch > CloudWatch Logs > Log groups > /aws-glue/python-jobs/output > [Switch to the original interface.](#)  
jr\_029c64f34edd6d06f31b9228125c409b1519ddaa854f0adb3aa3ce7a7e6ec28f

**Log events** Actions

Filter events  Clear 1m 30m 1h 12h

Timestamp	Message
	There are older events to load. <a href="#">Load more.</a>
2020-08-17T10:45:31.378+03:00	<pre> Considering file without prefix as a python extra file s3://movieswalker/jobs 1 2 3 4 5 </pre>
	No newer events at this moment. <a href="#">Auto retry paused.</a> <a href="#">Resume</a>

# Give Glue user access to S3 bucket

If you have run any of our other tutorials, like [running a crawler](#) or [joining tables](#), then you might already have the **AWSGlueServiceRole-S3IAMRole**. What's important for running a Glue job is that the role has access to the S3 bucket where the Python script is stored.

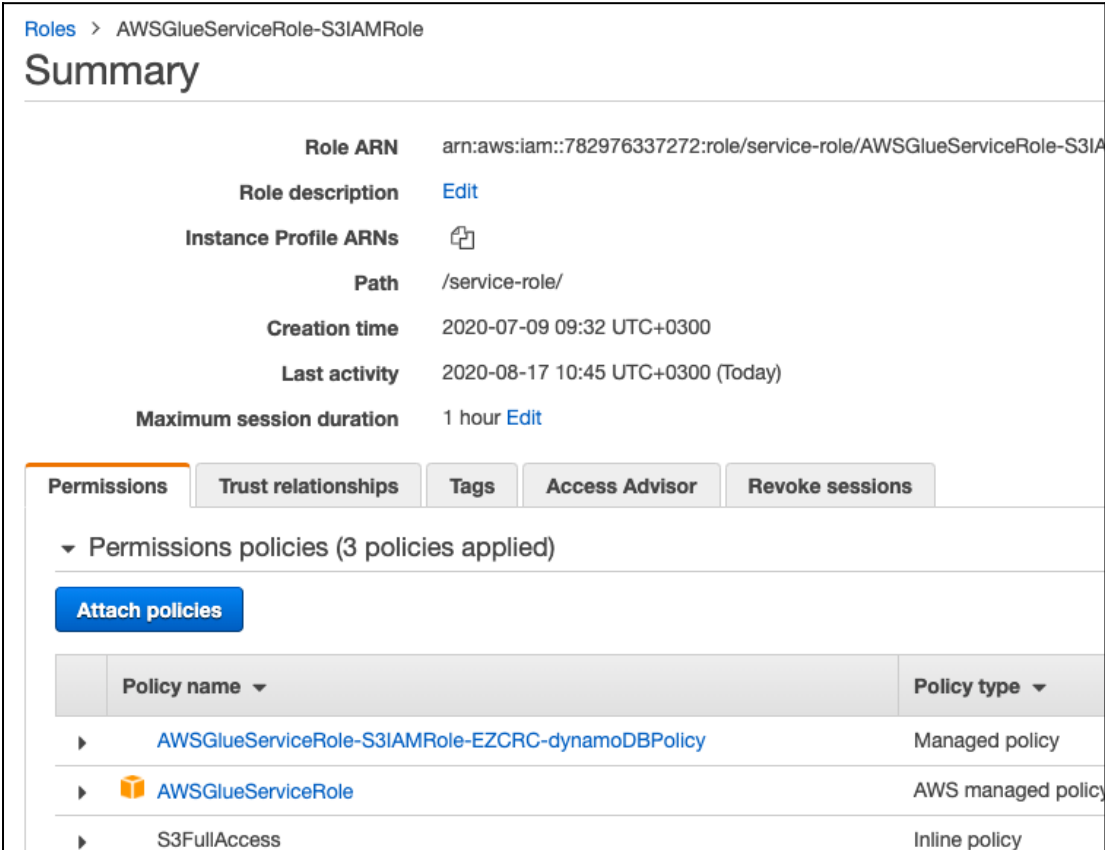
In this example, I added that manually using the JSON Editor in the IAM roles screen and pasted in this policy:

```
{
  "Version": "2012-10-17",
  "Statement":
}
```

If you don't do this, or do it incorrectly, you will get this error:

```
File "/usr/local/lib/python3.6/site-packages/botocore/client.py", line 661,
in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (403) when calling the
HeadObject operation: Forbidden
```

Here we show that the user has the **AWSGlueServiceRole** policy and the S3 policy we just added in the **AWSGlueServiceRole-S3IAMRole** role. That, of course, must be attached to your IAM user.



The screenshot shows the AWS IAM console interface for the role **AWSGlueServiceRole-S3IAMRole**. The **Summary** tab is active, displaying the following details:

- Role ARN:** `arn:aws:iam::782976337272:role/service-role/AWSGlueServiceRole-S3IAMRole`
- Role description:** [Edit](#)
- Instance Profile ARNs:** [+](#)
- Path:** `/service-role/`
- Creation time:** 2020-07-09 09:32 UTC+0300
- Last activity:** 2020-08-17 10:45 UTC+0300 (Today)
- Maximum session duration:** 1 hour [Edit](#)

Below the summary, the **Permissions** tab is selected, showing that 3 policies are applied. A table lists these policies:

Policy name	Policy type
<a href="#">AWSGlueServiceRole-S3IAMRole-EZCRC-dynamoDBPolicy</a>	Managed policy
<a href="#">AWSGlueServiceRole</a>	AWS managed policy
<a href="#">S3FullAccess</a>	Inline policy

## Additional resources

Explore these resources:

- [AWS Glue ETL Transformations](#)
- [BMC Machine Learning & Big Data Blog](#)
- [Apache Spark Guide](#)
- [AWS Guide](#)