







## Table Of Contents

<i>Create Lambda For S3 Bucket(The Lambda Name SHOULD Contain Gluetriggerlambda As Part Of The Name). .....</i>	<i>2</i>
<i>Creating Cloud 9 Environment.....</i>	<i>3</i>
<i>Create S3 Bucket.....</i>	<i>4</i>
<i>Adding Trigger To S3 Bucket .....</i>	<i>5</i>
<i>Creating The Crawler .....</i>	<i>7</i>
<i>Cloud 9 Python Scripts To Push Files To S3 .....</i>	<i>11</i>
<i>Aws Glue Jobs.....</i>	<i>15</i>
<i>Adding Lambdas To Invoke Glue Jobs From Crawler .....</i>	<i>18</i>
<i>Create Red Shift Cluster .....</i>	<i>22</i>

## Create Lambda For S3 Bucket(The Lambda Name SHOULD Contain Gluetriggerlambda As Part Of The Name).

1. Go to Lambda->Create Function and choose the following settings. The IAM “LambdaEverything” role has the following policies selected

	Policy name ▼
▶	 AmazonS3FullAccess
▶	 AWSGlueServiceRole
▶	 AmazonRedshiftFullAccess
▶	 AmazonS3ReadOnlyAccess
▶	 AWSGlueConsoleFullAccess
▶	 AmazonRedshiftDataFullAccess

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  

Python 3.9 ▼


**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

**▼ Change default execution role**

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).  

☐ Create a new role with basic Lambda permissions  
☒ Use an existing role  
☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  

▼ 

  
[View the LambdaEverything role on the IAM console.](#)

2. Once the function is created, add the following code and click Deploy.

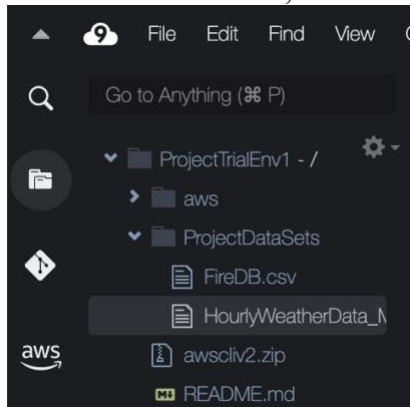
```
import json
import boto3
client=boto3.client('glue')
def lambda_handler(event, context):
    response = client.start_crawler(Name='FireAndWeatherCrawler')
    print(json.dumps(response, indent=4))
```

## Creating Cloud 9 Environment

1. create a cloud9 environment with ec2 instance with all default configurations

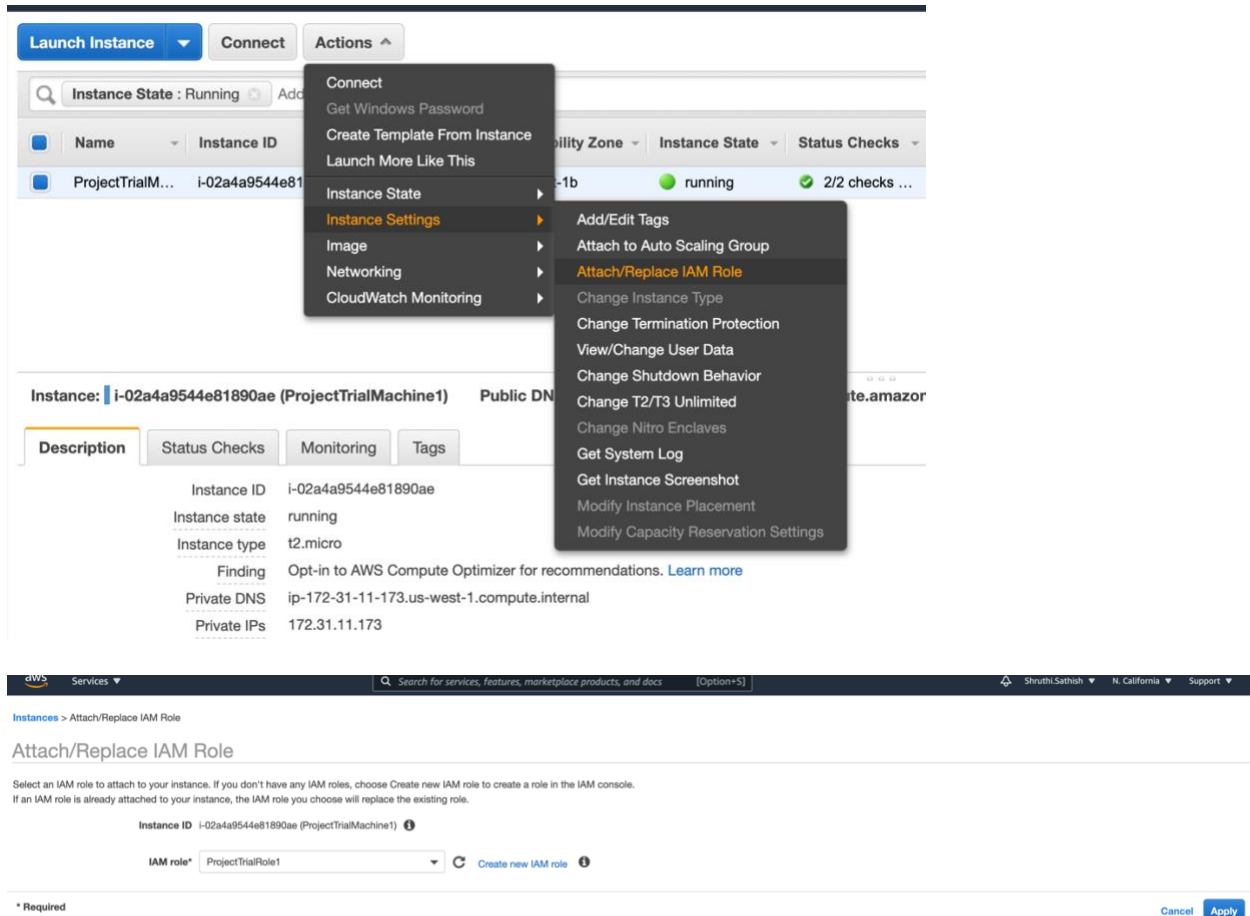
Go to EC2 instances and change name to “DataAvengersEC2”

2. In the cloud9 IDE, create a new folder “projectDataSets” and upload the CSV files



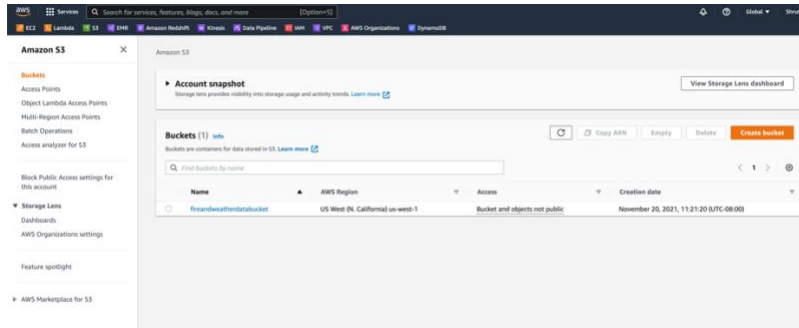
3. Create IAM Role to Allow EC2 instances to call AWS services on your behalf with full access to [AmazonS3FullAccess](#), [AWSLambda\\_FullAccess](#) and [AWSGlueServiceRole](#)

4. Attach this AWS role to the ec2 instance

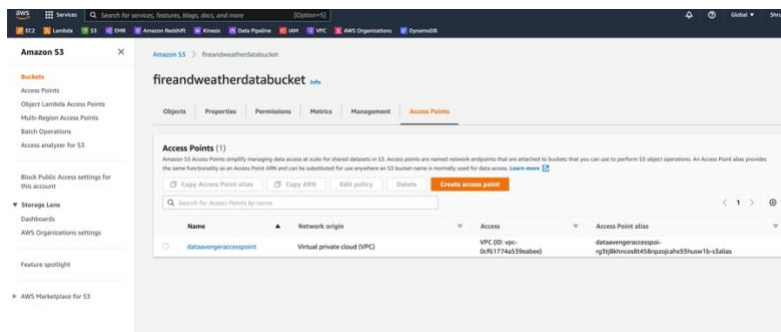
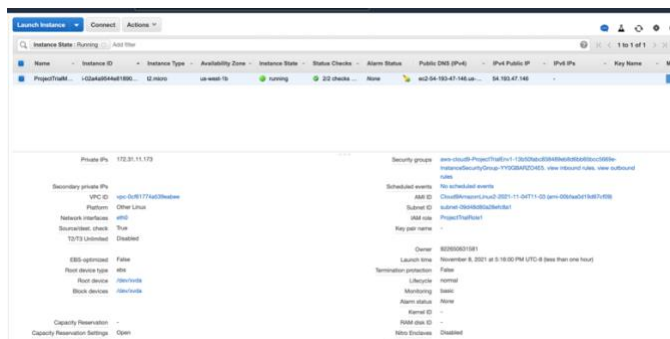
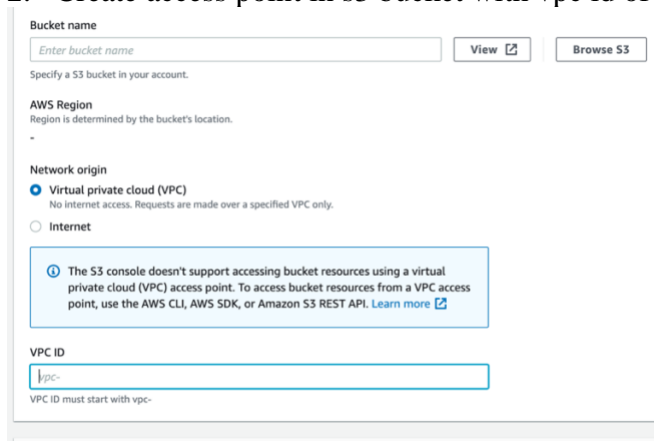


# Create S3 Bucket

## 1. Create s3 bucket



## 2. Create access point in s3 bucket with vpc id of the ec2 instance



## Adding Trigger To S3 Bucket


### 1. Go to Properties Tab

fireandweatherdatabucket [Info](#)

Objects **Properties** Permissions Metrics Management Access Points

#### Bucket overview

AWS Region  
US West (N. California) us-west-1

Amazon Resource Name (ARN)  
 arn:aws:s3:::fireandweatherdatabucket

Creation date  
November 1, 2019

### 2. Scroll down to Event Notification

#### Event notifications (0)

Send a notification when specific events occur in your bucket. [Learn more](#)

Edit

Delete

Create event notification

	Name	Event types	Filters	Destination type	Destination
--	------	-------------	---------	------------------	-------------

No event notifications

Choose Create event notification to be notified when a specific event occurs.

Create event notification

### 3. Create event notification. Select All object create events

#### General configuration

##### Event name

StartCrawler

Event name can contain up to 255 characters.

##### Prefix - optional

Limit the notifications to objects with key starting with specified characters.

images/

##### Suffix - optional

Limit the notifications to objects with key ending with specified characters.

.jpg

#### Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

##### Object creation

☒ All object create events  
s3:ObjectCreated:\*

☐ Put  
s3:ObjectCreated:Put

☐ Post  
s3:ObjectCreated:Post

☐ Copy  
s3:ObjectCreated:Copy

☐ Multipart upload completed  
s3:ObjectCreated:CompleteMultipartUpload

4. Choose destination as Lambda and select the lambda function. Save the changes

The screenshot shows the 'Destination' configuration page in the Amazon Lambda console. At the top, there is a blue information box stating: 'Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. [Learn more](#)'. Below this, the 'Destination' section is active, showing 'Lambda function' as the selected option. Under 'Specify Lambda function', 'Choose from your Lambda functions' is selected. The 'Lambda function' dropdown menu shows 'DA-GlueTriggerLambda-StartCrawler'. At the bottom right, there are 'Cancel' and 'Save changes' buttons.

5. Now when you open the Lambda function again, the trigger will be added  
DA-GlueTriggerLambda-StartCrawler

The screenshot shows the 'Function overview' page for 'DA-GlueTriggerLambda-StartCrawler'. The function has one trigger of type 'S3'. The 'Layers' section shows '(0)'. The 'Description' field is empty. The 'Last modified' timestamp is '20 seconds ago'. The 'Function ARN' is 'arn:aws:lambda:us-west-1:922650631581:function:DA-GlueTriggerLambda-StartCrawler'. Below the overview, the 'Code source' tab is selected, showing the code for 'lambda\_function.py'. The code is as follows:

```
1 import json
2 import boto3
3 client = boto3.client('glue')
4 def lambda_handler(event, context):
5     response = client.start_crawler(Name='FirstAndWeatherCrawler')
6     print(json.dumps(response, indent=4))
7
```

# Creating The Crawler

## 1. Create IAM role to access Glue

Budgets

Certificate Manager

Chime

CloudFormation

CloudHSM

CloudTrail

CloudWatch Alarms

CloudWatch Application Insights

EC2

EC2 - Fleet

EC2 Auto Scaling

EC2 Image Builder

EKS

Homecode

IAM Access Analyzer

Incident Manager

Inspector

IoT

OpsWorks

Panorama

Personalize

Purchase Orders

QLDB

Transfer

Trusted Advisor

VPC

WorkLink

WorkMail

Select your use case

Glue

Allows Glue to call AWS services on your behalf.

\* Required

Cancel

Next: Permissions

### Create role

1 2 3 4

#### ▼ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

Filter policies  Showing 8 results

	Policy name	Used as
<input checked="" type="checkbox"/>	AWSGlueConsoleFullAccess	Permissions policy (2)
<input type="checkbox"/>	AWSGlueConsoleSageMakerNotebookFullAccess	None
<input type="checkbox"/>	AwsGlueDataBrewFullAccessPolicy	Permissions policy (1)
<input type="checkbox"/>	AWSGlueDataBrewServiceRole	None
<input type="checkbox"/>	AWSGlueSchemaRegistryFullAccess	None
<input type="checkbox"/>	AWSGlueSchemaRegistryReadOnlyAccess	None
<input type="checkbox"/>	AWSGlueServiceNotebookRole	None
<input checked="" type="checkbox"/>	AWSGlueServiceRole	Permissions policy (1)

#### ▶ Set permissions boundary

\* Required

Cancel Previous Next: Tags

### Create role

1 2 3 4

#### ▼ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

Filter policies  Showing 10 results

	Policy name	Used as
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	None
<input checked="" type="checkbox"/>	AmazonS3FullAccess	Permissions policy (3)
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRolePolicy	None
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	None
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	None
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	Permissions policy (1)
<input type="checkbox"/>	AWSLambdaS3ExecutionRole-787a04af-b84c-49c7-a6a0-cc4a52a9a7b4	Permissions policy (1)
<input type="checkbox"/>	IVSRecordToS3	None

#### ▶ Set permissions boundary

2. Create Glue crawler with IAM and create a new DB to create tables after the data is crawled  
AWS Glue -> Crawlers -> Add Crawler

**Add crawler**

**FireAndWeatherCrawl**  
or  
**FireAndWeatherCrawl**

**Crawler source type**

☐ Data store  
☐ IAM Role  
☐ Schedule  
☐ Output  
☐ Remove all steps

**Add information about your crawler**

Crawler name

• Tags, description, security configuration, and classifiers (optional)

[Next](#)

**Add crawler**

**FireAndWeatherCrawl**  
or  
**FireAndWeatherCrawl**

**Crawler source type**

☐ Data store  
☐ IAM Role  
☐ Schedule  
☐ Output  
☐ Remove all steps

**Specify crawler source type**

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

**Crawler source type**

☒ Data stores  
☐ Existing catalog tables

**Repeat crawls of S3 data stores**

☒ Crawl all folders  
Crawl all folders again with every subsequent crawl.

☐ Crawl new folders only  
Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

☐ Crawl changed folders identified by Amazon S3 Event Notifications  
Rely on Amazon S3 events to control what folders to crawl.

[Back](#) [Next](#)

Include path to s3 bucket

**Add crawler**

**FireAndWeatherCrawl**  
or  
**FireAndWeatherCrawl**

**Crawler source type**

☐ Data store  
☐ IAM Role  
☐ Schedule  
☐ Output  
☐ Remove all steps

**Add a data store**

Choose a data store

Connection

Optionally include a fallback connection to use with this S3 target. Note that each crawler is limited to one fallback connection for any future S3 targets will also use the same connection (or none, if left blank).

[Add connection](#)

**Crawl data in**

☒ Specified path in my account  
☐ Specified path in another account

**Include path**

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

**Sample size (optional)**

You have seen the number of files in each folder to be crawled. If not set, all the files are crawled.

• **Exclude patterns (optional)**

[Back](#) [Next](#)

**Add crawler**

**FireAndWeatherCrawl**  
or  
**FireAndWeatherCrawl**

**Crawler source type**

☐ Data store  
☐ IAM Role  
☐ Schedule  
☐ Output  
☐ Remove all steps

**Choose an IAM role**

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

☐ Update a policy in an IAM role  
☒ Choose an existing IAM role  
☐ Create an IAM role

**IAM role**

This role must provide permissions similar to the AWS managed policy, **AWSGlueServiceRole**, plus access to your data stores.

• s3://fireandweatherdatabucket

You can also create an IAM role on the [IAM console](#).

[Back](#) [Next](#)



### 3. Add new Database

**Add database**

Database name  
fireweatherdb

Description and location (optional)

Resource link name  
Enter resource link name

Shared database suggestions  
Choose a database to autofill form

Shared database  
Enter database to link to

Shared database owner account ID  
Enter an AWS account ID

Create

**AWS Glue**

Data catalog

Databases

Tables

Connections

**Crawlers**

Classifiers

**Crawlers** A crawler connects to a data store, progresses through a prioritized list c

Add crawler Run crawler Action Filter by tags and attributes

<input type="checkbox"/>	Name	Schedule
<input type="checkbox"/>	FireAndWeatherCrawler	

### 4. Open the Lambda Function for crawler and go to Test Tab Select event as “S3-Put” from drop down and click test

**Test event** Format Save changes Test

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event Saved event

Template  
s3-put

Name  
MyEventName

```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-east-1",
7       "eventTime": "1970-01-01T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
9       "userIdentity": {
10        "principalId": "EXAMPLE"
11      },
12       "requestParameters": {
13        "sourceIPAddress": "127.0.0.1"
14      },
15       "responseElements": {
16        "x-amz-request-id": "EXAMPLE123456789",
17        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawsome/mnopqrstuvwxyzABCDEFGH"
18      },
19       "s3": {
20        "s3SchemaVersion": "1.0",
21        "configurationId": "testConfigRule",
22        "bucket": {
23          "name": "example-bucket",
24          "ownerIdentity": {
25            "principalId": "EXAMPLE"
26          },
27          "arn": "arn:aws:s3:::example-bucket"
```

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

null

Summary

Code SHA-256	Request ID
h7QWCyEhugZT+Q9ChCP4o42pTBDDmZ9Wo3JXQ=	bd6e7344-ada9-454e-bbd7-05cf18b65321
Duration	Billed duration
2458.03 ms	2439 ms
Resources configured	Max memory used
128 MB	68 MB

Log output

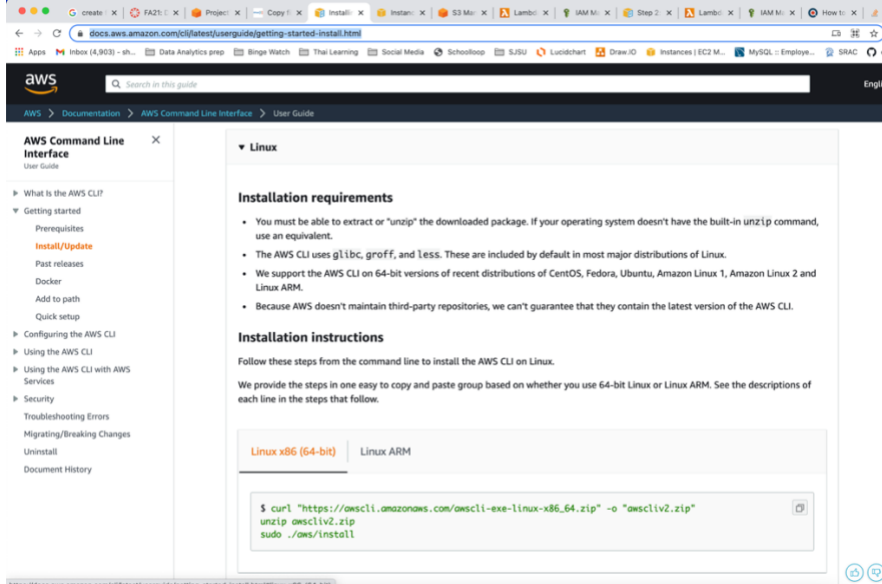
The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

```
START RequestId: bd6e7344-ada9-454e-bbd7-05cf18b65321 Version: $LATEST
{
  "ResponseMetadata": {
    "RequestId": "218eaf09-e1ad-4e84-ba4e-4d8a831b21a5",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Wed, 24 Nov 2021 20:15:27 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "2",
      "connection": "keep-alive",
```

5. This means that the trigger is working properly

## Cloud 9 Python Scripts To Push Files To S3

1. Install latest version of cli on EC2 using cloude9 IDE more details at the link <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>



2. Use the cli aws copy command to copy files from ec2 to s3 bucket
  - aws s3 ls s3://fireandweatherbucket

### SKIP THESE NEXT 2 STEPS IN RED

- aws s3 cp ProjectDataSets/FireDB.csv s3:// fireandweatherbucket/Fires
- aws s3 cp ProjectDataSets/HourlyWeatherData\_Merged.csv s3:// fireandweatherbucket/Weather

```
inflating: aws/dist/docutils/parsers/rst/include/isoLat2.txt
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/LICENSE.BSD
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/INSTALLER
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/RECORD
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/LICENSE.PSF
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/AUTHORS.rst
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/top_level.txt
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/METADATA
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/LICENSE.APACHE
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/LICENSE
inflating: aws/dist/cryptography-3.3.2-py3.8.egg-info/WHEEL
creating: aws/dist/cryptography/hazmat/
creating: aws/dist/cryptography/hazmat/bindings/
inflating: aws/dist/cryptography/hazmat/bindings/_openssl.abi3.so
ec2-user:~/environment $ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ec2-user:~/environment $ aws --version
aws-cli/2.3.4 Python/3.8.8 Linux/4.14.248-189.473.amzn2.x86_64 exe/x86_64.amzn.2 prompt/off
ec2-user:~/environment $ aws s3 ls s3://projecttrialbucket1
ec2-user:~/environment $ aws s3 cp /ProjectDataSets/FireDB.csv s3://projecttrialbucket1

The user-provided path /ProjectDataSets/FireDB.csv does not exist.
ec2-user:~/environment $ aws s3 cp ~/ProjectDataSets/FireDB.csv s3://projecttrialbucket1

The user-provided path /home/ec2-user/ProjectDataSets/FireDB.csv does not exist.
ec2-user:~/environment $ aws s3 cp ProjectDataSets/FireDB.csv s3://projecttrialbucket1
upload: ProjectDataSets/FireDB.csv to s3://projecttrialbucket1/FireDB.csv
ec2-user:~/environment $ aws s3 cp ProjectDataSets/HourlyWeatherData_Merged.csv s3://projecttrialbucket1
upload: ProjectDataSets/HourlyWeatherData_Merged.csv to s3://projecttrialbucket1/HourlyWeatherData_Merged.csv
ec2-user:~/environment $
```

### 3. Using PYTHON to move files to S3 Bucket

<https://docs.aws.amazon.com/cloud9/latest/user-guide/sample-python.html#sample-python-sdk>

#### Install Python

```
ec2-user:~/environment $ sudo yum -y update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
234 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package aws-cfn-bootstrap.noarch 0:2.0-6.amzn2 will be updated
--> Package aws-cfn-bootstrap.noarch 0:2.0-9.amzn2 will be an update
--> Package containerd.x86_64 0:1.4.6-3.amzn2 will be updated
--> Package containerd.x86_64 0:1.4.6-7.amzn2 will be an update
--> Package docker.x86_64 0:20.10.7-3.amzn2 will be updated
--> Package docker.x86_64 0:20.10.7-5.amzn2 will be an update
--> Package ec2-instance-connect.noarch 0:1.1-14.amzn2 will be updated
--> Package ec2-instance-connect.noarch 0:1.1-15.amzn2 will be an update
--> Package glibc.x86_64 0:2.26-55.amzn2 will be updated
--> Package glibc.x86_64 0:2.26-56.amzn2 will be an update
--> Package glibc-all-langpacks.x86_64 0:2.26-55.amzn2 will be updated
--> Package glibc-all-langpacks.x86_64 0:2.26-56.amzn2 will be an update
--> Package glibc-common.x86_64 0:2.26-55.amzn2 will be updated
--> Package glibc-common.x86_64 0:2.26-56.amzn2 will be an update
--> Package glibc-devel.x86_64 0:2.26-55.amzn2 will be updated
--> Package glibc-devel.x86_64 0:2.26-56.amzn2 will be an update
--> Package glibc-headers.x86_64 0:2.26-55.amzn2 will be updated
--> Package glibc-headers.x86_64 0:2.26-56.amzn2 will be an update
--> Package glibc-locale-source.x86_64 0:2.26-55.amzn2 will be updated
```

```
Complete!
ec2-user:~/environment $ sudo yum -y install python3
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
234 packages excluded due to repository priority protections
Package python3-3.7.10-1.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
ec2-user:~/environment $
```

### 4. Install and configure the AWS SDK for Python (Boto3)

#### Install pip

```
ec2-user:~/environment $ curl -O https://bootstrap.pypa.io/get-pip.py # Get the install script.
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 2108k 100 2108k 0 0 21.0M 0 --:--:-- --:--:-- --:--:-- 21.0M
ec2-user:~/environment $ sudo python3 get-pip.py # Install pip for Python 3.6.
sudo: python3: command not found
ec2-user:~/environment $ python -m pip --version # Verify pip is installed.
pip 20.2.2 from /usr/lib/python3.7/site-packages/pip (python 3.7)
ec2-user:~/environment $ rm get-pip.py # Delete the install script.
ec2-user:~/environment $ python -m pip --version
pip 20.2.2 from /usr/lib/python3.7/site-packages/pip (python 3.7)
ec2-user:~/environment $
```

#### Install the AWS SDK for Python (Boto3)

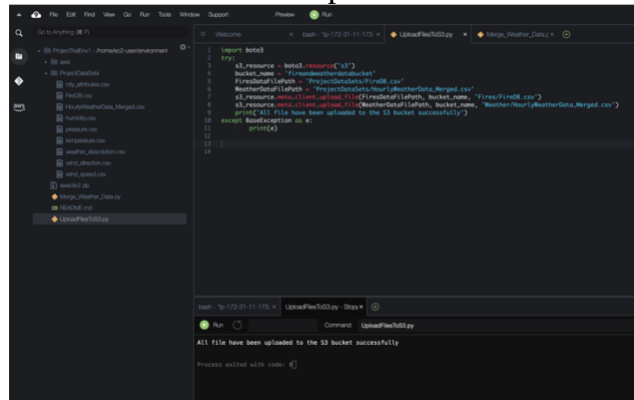
```
ec2-user:~/environment $ sudo python3 -m pip install boto3
WARNING: Running pip install with root privileges is generally not a good idea. Try 'python3 -m pip install --user' instead.
Collecting boto3
  Downloading boto3-1.20.10-py3-none-any.whl (131 kB)
    | 131 kB 29.0 MB/s
Collecting botocore<1.24.0,>=1.23.10
  Downloading botocore-1.23.10-py3-none-any.whl (8.2 MB)
    | 8.2 MB 14.1 MB/s
Collecting s3transfer<0.6.0,>=0.5.0
  Downloading s3transfer-0.5.0-py3-none-any.whl (79 kB)
    | 79 kB 14.7 MB/s
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.7/site-packages (from boto3) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/local/lib/python3.7/site-packages (from botocore<1.24.0,>=1.23.10->boto3) (1.26.7)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/site-packages (from botocore<1.24.0,>=1.23.10->boto3) (2.8.2)
Requirement already satisfied: six<1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.24.0,>=1.23.10->boto3) (1.16.0)
Installing collected packages: botocore, s3transfer, boto3
  Attempting uninstall: botocore
    Found existing installation: botocore 1.22.10
    Uninstalling botocore-1.22.10:
      Successfully uninstalled botocore-1.22.10
Successfully installed boto3-1.20.10 botocore-1.23.10 s3transfer-0.5.0
ec2-user:~/environment $
```

```
ec2-user:~/environment $ python -m pip show boto3
Name: boto3
Version: 1.20.10
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email: None
License: Apache License 2.0
Location: /usr/local/lib/python3.7/site-packages
Requires: jmespath, s3transfer, botocore
Required-by:
ec2-user:~/environment $
```

## 5. Install Pandas

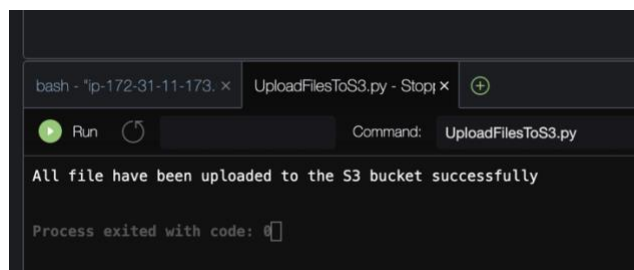
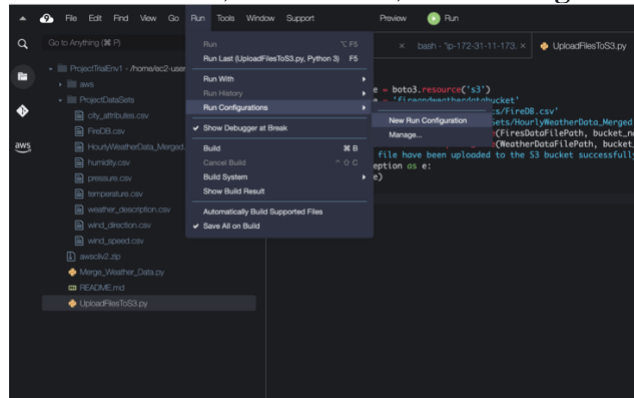
```
Complete!
ec2-user:~/environment $ sudo pip3 install pandas
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting pandas
  Downloading pandas-1.3.4-cp37m-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.3 MB)
    |#####| 11.3 MB 50 kB/s
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/site-packages (from pandas) (2021.3)
Collecting numpy>=1.17.3; platform_machine != "aarch64" and platform_machine != "arm64" and python_version < "3.10"
  Downloading numpy-1.21.4-cp37m-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)
    |#####| 15.7 MB 62 kB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Installing collected packages: numpy, pandas
  WARNING: The scripts f2py, f2py3 and f2py3.7 are installed in '/usr/local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.21.4 pandas-1.3.4
ec2-user:~/environment $
```

## 6. Create a file called UploadFilesToS3 and write the following code

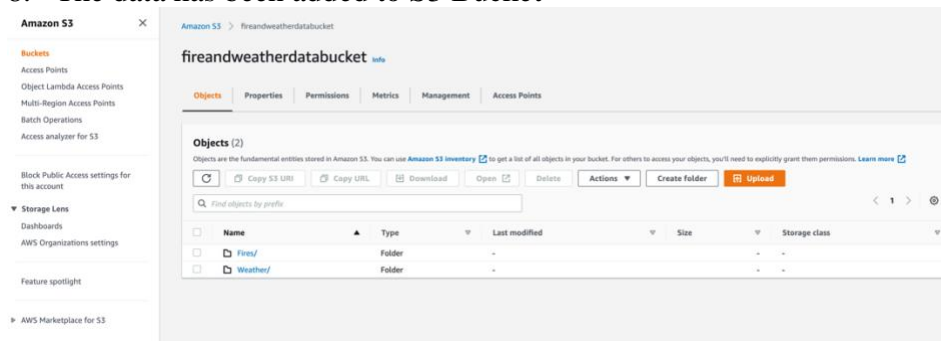


## 7. Execute the python script

On the menu bar, choose **Run, Run Configurations, New Run Configuration**

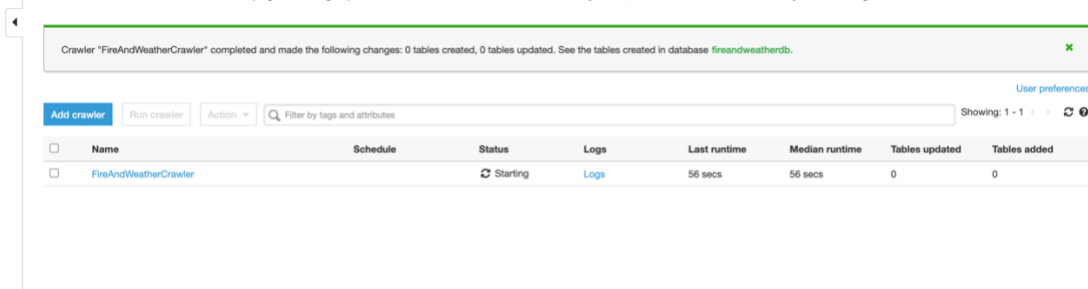


## 8. The data has been added to S3 Bucket



## 9. Go to the AWS Glue and Check if the crawler has started. The crawler should run automatically once the file is uploaded into the S3 bucket

Crawlers A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.



## 10. Once the crawler stops, 2 new tables will be created in the DB



# Aws Glue Jobs

## 1. Create Glue job with following Configurations

AWS Glue -> ETL -> Jobs -> Add Job

**Add job**

**Configure the job properties**

**Job properties**

- ☒ Job properties
- ☐ Data source
- ☐ Transform type
- ☐ Data target
- ☐ Schema

**Name**  
firetaglujob

**IAM role**  
AWSProjectFullAccess

**Type**  
Spark

**Glue version**  
Spark 2.4, Python 3 (Glue Version 2.0)

**This job runs**  
☒ A proposed script generated by AWS Glue

**Script file name**  
firetaglujob

**S3 path where the script is stored**  
s3://aws-glue-scripts-922650631581-us-west-1/root

**Temporary directory**  
s3://aws-glue-temporary-922650631581-us-west-1/root

## 2. Select Fires as the datasource

**Add job**

**Choose a data source**

☒ Job properties

**Data source**

- ☒ Data source
- ☐ Transform type
- ☐ Data target
- ☐ Schema

Filter by attributes or search by keyword

Name	Database	Location	Classification
fires	fireweatherdb	s3://fireweatherdatabasebucket/Fires/	csv
weather	fireweatherdb	s3://fireweatherdatabasebucket/Weather/	csv

Showing: 1 - 2

**Choose a transform type**

☒ **Change schema**  
Change schema of your source data and create a new target dataset

☐ **Find matching records**  
Use machine learning to find matching records within your source data

**Back** **Next**

## 3. Choose target as S3, format Parquet and Target Path as /Fires/Parquet

**Choose a data target**

☒ Create tables in your data target

☐ Use tables in the data catalog and update your data target

**Data store**  
Amazon S3

**Format**  
Parquet

**Connection**  
Select one

**Target path**  
s3://fireweatherdatabasebucket/Fires/Parquet

**Back** **Next**

## 4. Remove the ObjectID and Shape columns from Schema

Output Schema Definition

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source			Target		
Column name	Data type	Map to target	Column name	Data type	
objectId	string	-	fod_id	long	✕ ↓ ↑
fod_id	bigint	fod_id	fpa_id	string	✕ ↓ ↑
fpa_id	string	fpa_id	source_system_type	string	✕ ↓ ↑
source_system_type	string	source_system_type	source_system	string	✕ ↓ ↑
source_system	string	source_system	nwcg_reporting_agency	string	✕ ↓ ↑
nwcg_reporting_agency	string	nwcg_reporting_agency	nwcg_reporting_unit_id	string	✕ ↓ ↑
nwcg_reporting_unit_id	string	nwcg_reporting_unit_id	nwcg_reporting_unit_name	string	✕ ↓ ↑
nwcg_reporting_unit_name	string	nwcg_reporting_unit_name	source_reporting_unit	long	✕ ↓ ↑
source_reporting_unit	bigint	source_reporting_unit	source_reporting_unit_name	string	✕ ↓ ↑
source_reporting_unit_name	string	source_reporting_unit_name	local_fire_report_id	long	✕ ↓ ↑
local_fire_report_id	bigint	local_fire_report_id	local_incident_id	string	✕ ↓ ↑
local_incident_id	string	local_incident_id	fire_code	string	✕ ↓ ↑
fire_code	string	fire_code	fire_name	string	✕ ↓ ↑
fire_name	string	fire_name	ics_209_incident_number	string	✕ ↓ ↑

## 5. Save the script and run the job

Add job Action Filter by tags and attributes

Name	Type	ETL language
firedataglujob	Spark	python

History								
View run metrics Rewind job bookmark								
Run ID	Retry attempt	Run status	Error	Output	Logs	Error logs	Glue version	Maximum capacity
j_r_de63b2e8b551...	-	Running			Logs	Error logs	2.0	10

## 6. The parquet files will be saved in the S3 bucket

Amazon S3

Amazon S3 > fireandweatherdatabucket > Files/ > Parquet/

Parquet/ Copy S3 URI

Objects (11)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
part-00000-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:27 (UTC-08:00)	8.0 MB	Standard
part-00001-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:30 (UTC-08:00)	8.8 MB	Standard
part-00002-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:28 (UTC-08:00)	5.8 MB	Standard
part-00003-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:28 (UTC-08:00)	6.3 MB	Standard
part-00004-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:28 (UTC-08:00)	5.4 MB	Standard
part-00005-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:28 (UTC-08:00)	5.7 MB	Standard
part-00006-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:29 (UTC-08:00)	8.0 MB	Standard
part-00007-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:28 (UTC-08:00)	8.2 MB	Standard
part-00008-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:04 (UTC-08:00)	9.0 MB	Standard
part-00009-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:48:04 (UTC-08:00)	8.7 MB	Standard
part-00010-SF561020-e1f2-433a-857e-9b5e5716c04d-c000.snappy.parquet	parquet	November 20, 2021, 17:47:57 (UTC-08:00)	1.2 MB	Standard



## 7. Create another Glue job for weather data

**Add job**

Job properties  
Data source  
Transform type  
Data target  
Schema

**Configure the job properties**

Name: weatheringgluejob

JAR role: AWSGlueElasticMapReduce

Type: Spark

Glue version: Spark 3.4, Python 3 (Glue Version 2.0)

This job name:  
☒ A proposed script generated by AWS Glue  
☐ An existing script that you provide  
☐ A new script to be authored by you

Script file name: weatheringgluejob

S3 path where the script is stored: s3://new-glue-script-62260631081-us-west-1-root

Temporary directory: s3://new-glue-temporary-62260631081-us-west-1-root

**Choose a data source**

Filter by attributes or search by keyword

Name	Database	Location	Classification
free	freeweatherdb	s3://freeweatherdatabucket/Free/	csv
weather	freeweatherdb	s3://freeweatherdatabucket/Weather/	csv

Showing 1 - 2

**Choose a data target**

☒ Create tables in your data target  
☐ Use tables in the data catalog and update your data target

Data store: Amazon S3

Format: Parquet

Connection: - Select one -

Add connection

Target path: s3://freeweatherdatabucket/Weather/Parquet

Back Next

**Add job**

Job properties  
Data source  
Transform type  
Change schema  
Data target  
s3://freeweather...  
Schema

**Output Schema Definition**

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source	Target
Column name	Column name
id	id
timestamp	timestamp
locations	locations
humidity	humidity
pressure	pressure
temperature	temperature
wind direction	wind direction
wind speed	wind speed
weather description	weather description

## 8. Save and run the job. New parquet files for weather data will be added to S3 bucket

**Amazon S3**

Buckets  
Access Points  
Object Lambda Access Points  
Multi-Region Access Points  
Batch Operations  
Access analyzer for S3

Block Public Access settings for this account

Storage Lens  
Elasticsearch  
AWS Organizations settings

Feature spotlight

► AWS Marketplace for S3

**Amazon S3** > freeweatherdatabucket > Weather/ > Parquet/

Parquet/

Copy S3 URL

Objects (2)

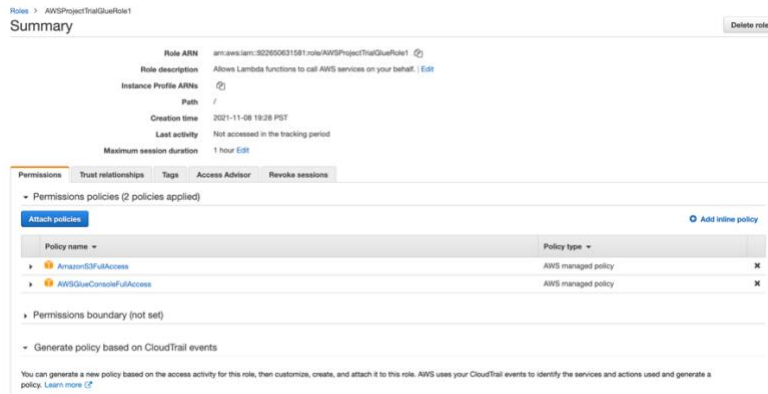
Find objects by prefix

Name	Type	Last modified	Size	Storage class
s3://0000-244801v-bd84-aws-us-east-1/s3-us-east-1-glue-jobs-0001-empy-parquet	parquet	November 20, 2021, 18:07:33 GMT-08:00	11.8 MB	Standard
s3://0001-244801v-bd84-aws-us-east-1/s3-us-east-1-glue-jobs-0001-empy-parquet	parquet	November 20, 2021, 18:07:33 GMT-08:00	8.7 MB	Standard

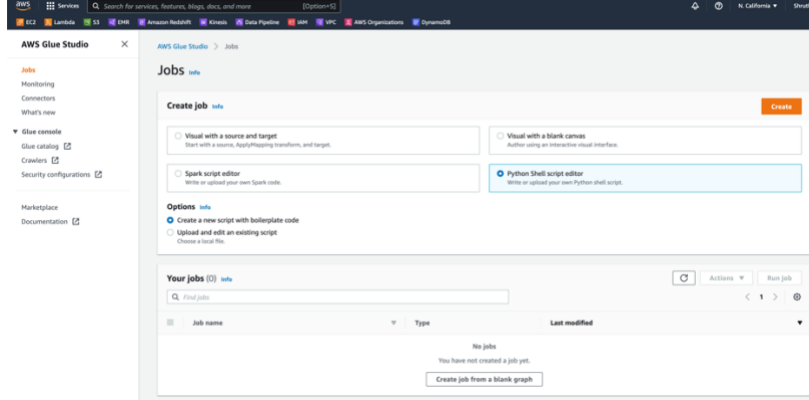
## Adding Lambdas To Invoke Glue Jobs From Crawler

<https://aws.amazon.com/premiumsupport/knowledge-center/start-glue-job-crawler-completes-lambda/>

### 1. Create IAM AWS Role for Lambda access with full access to S3 bucket and Glue console



### 2. Create Lambda function with IAM role and python script



Add the code

```
# Set up logging
```

```
import json
```

```
import os
```

```
import logging
```

```
logger = logging.getLogger()
```

```
logger.setLevel(logging.INFO)
```

```
# Import Boto 3 for AWS Glue
```

```
import boto3
```

```
client = boto3.client('glue')
```

```
# Variables for the job:
```

```
glueJobName = "firedatagluejob"
```

```
# Define Lambda function
```

```
def lambda_handler(event, context):
```

```
    logger.info('## INITIATED BY EVENT: ')
```

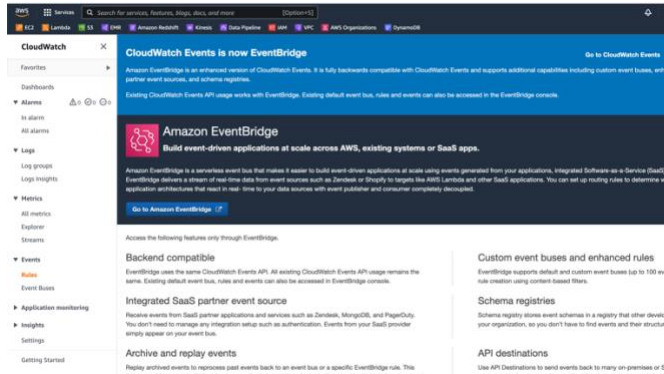
```

logger.info(event['detail'])
response = client.start_job_run(JobName = glueJobName)
logger.info('## STARTED GLUE JOB: ' + glueJobName)
logger.info('## GLUE JOB RUN ID: ' + response['JobRunId'])
#response1 = client.start_job_run(JobName = glueJobName2)
#logger.info('## STARTED GLUE JOB: ' + glueJobName2)
#logger.info('## GLUE JOB RUN ID: ' + response1['JobRunId'])
return response

```

The screenshot shows the AWS Lambda console with the 'Code source' tab selected. The code is a Python Lambda function that uses the boto3 client to start a Glue job run. The code includes logging for the job name and job run ID. The function is named 'TestCrawl' and is located in the 'TestCrawl' folder.

### 3. Go to cloud watch -> Events ->Rules ->Event Bridge ->Create Rule



Name and description

Name

TestCrawl

Rule with the name already exists on this event bus.

Maximum of 64 characters consisting of lower/upper case letters, -, ., \_

Description - optional

Enter description

Define pattern

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event pattern

Schedule

Event matching pattern

You can use pre-defined pattern provided by a service or create a custom pattern.

Pre-defined pattern by service

Custom pattern

Event pattern

Save

Cancel

Pre-defined event pattern

In Custom pattern add the code and change the crawler name to our crawler

```

{
  "detail-type": [

```

```

    "Glue Crawler State Change"
  ],
  "source": [
    "aws.glue"
  ],
  "detail": {
    "crawlerName": [
      "FireAndWeatherCrawler"
    ],
    "state": [
      "Succeeded"
    ]
  }
}

```

**Define pattern**

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☒ **Event pattern** Info  
Build a pattern to match events

☐ **Schedule** Info  
Invoke your targets on a schedule

**Event matching pattern**  
You can use pre-defined pattern provided by a service or create a custom pattern

☐ Pre-defined pattern by service

☒ **Custom pattern**

**Event pattern** Save Cancel

```

1 {
2   "detail-type": [
3     "Glue Crawler State Change"
4   ],
5   "source": [
6     "aws.glue"
7   ],
8   "detail": {
9     "crawlerName": [
10      "FireAndWeatherCrawler"
11    ],
12    "state": [
13      "Succeeded"
14    ]
15  }
16 }

```

[▶ Test event pattern](#)

#### 4. Add the lambda function in target and click create

**Select targets**

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

**Target** Remove  
Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

Lambda function ▼

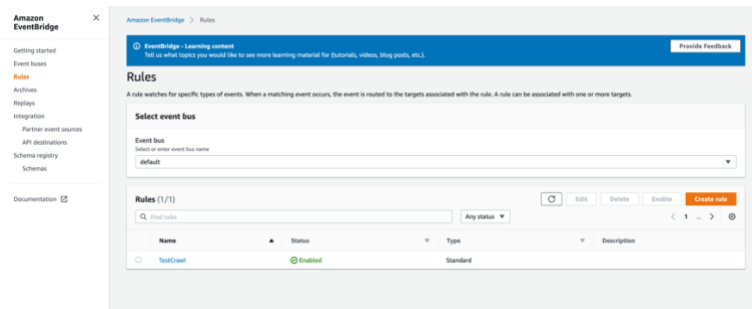
Function  
lambdaforglueaftercrawler ▼

[▶ Configure version/alias](#)

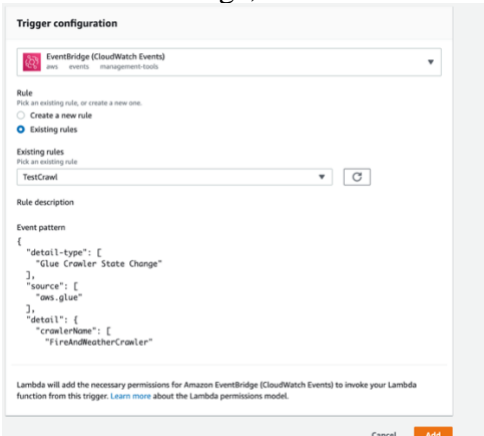
[▶ Configure input](#)

[▶ Retry policy and dead-letter queue](#)

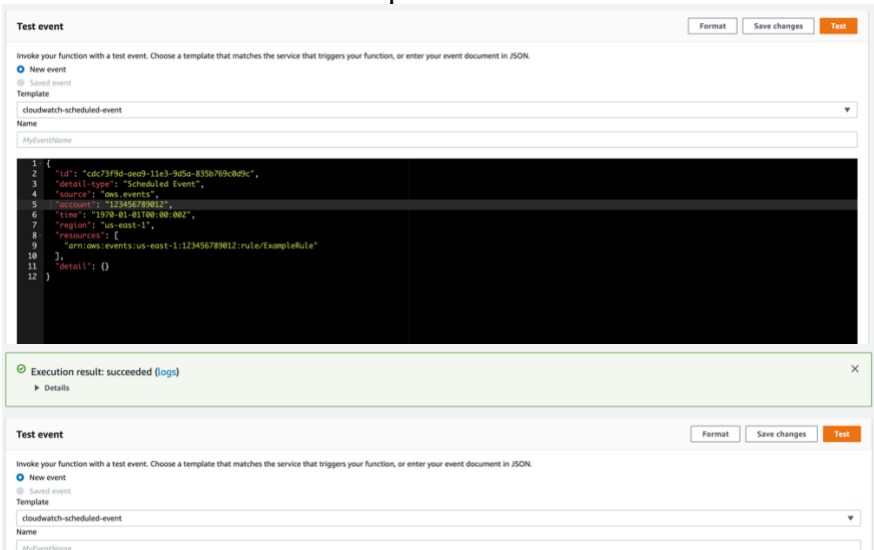
Add target



5. Go back to Lambda Function-> Add trigger  
Select EventBridge, name of the rule we created and click Add



6. In the Code tab of lambda function, click Deploy  
Go-to “Test” tab -> select template as “CloudWatch” and click test



7. Repeat the same and create another lambda function for “WeatherGlueJob”. But we can skip the rule creation. We can use the same rule we created for previous function. Everything else remains the same.

Now, go back to Glue and run the crawler. Once it stops running, go to jobs and check the history to check if the glue jobs are triggered.

# Create Red Shift Cluster

## 1. Create a RedShift Cluster

The screenshot shows the 'Create cluster' page in the Amazon Redshift console. The left sidebar contains navigation links for Dashboard, Clusters, Queries, Editor, Databases, Config, Marketplace, Advisor, and Alerts. The main content area is titled 'Create cluster' and includes a 'Cluster configuration' section. Under 'Cluster identifier', the text 'FireAndWeatherCluster' is entered. Below this, there are two options for 'What are you planning to use this cluster for?': 'Production' (selected) and 'Free trial'. The 'Production' option is described as 'Configure for fast and consistent performance at the best price.' The 'Free trial' option is described as 'Configure for learning about Amazon Redshift. This configuration is free for a limited time if your organization has never created an Amazon Redshift cluster.' Under 'Choose the size of the cluster', there are two buttons: 'I'll choose' and 'Help me choose'. Below this, the 'Node type' is set to 'ra3.4xlarge'. At the bottom, there is a section for 'AQUA (Advanced Query Accelerator)' with an 'Automatic' option selected.

## 2. Choose AWS User as the admin and password as DataAvengers2021

The screenshot shows the 'Database configurations' section in the Amazon Redshift console. It includes fields for 'Admin user name' and 'Admin user password'. The 'Admin user name' field contains 'awsuser'. Below this, there is a checkbox for 'Auto generate password' which is unchecked. The 'Admin user password' field is masked with asterisks. Below this, there is a checkbox for 'Show password' which is unchecked. The text below the password field states: 'Must be 8-64 characters long. Must contain at least one uppercase letter, one lowercase letter and one number. Can be any printable ASCII character except "/", "\*", "", or "@'".

## 3. In IAM roles, choose the one with full access to s3, glue and redshift and create cluster

The screenshot shows the 'Associate IAM roles' dialog in the Amazon Redshift console. It displays a list of IAM roles under the heading 'IAM roles (1/4)'. The roles listed are 'AmazonRedshift-CommandsAccessRole-20211120T013047', 'AWSServiceRoleForRedshift', 'myRedshiftRole', and 'RedshiftRoleEverything'. The 'RedshiftRoleEverything' role is selected with a checkbox. At the bottom of the dialog, there are two buttons: 'Cancel' and 'Associate IAM roles'.

4. In the Query Editor of amazon cluster. Click on connect to database and it will automatically connect to dev database as default

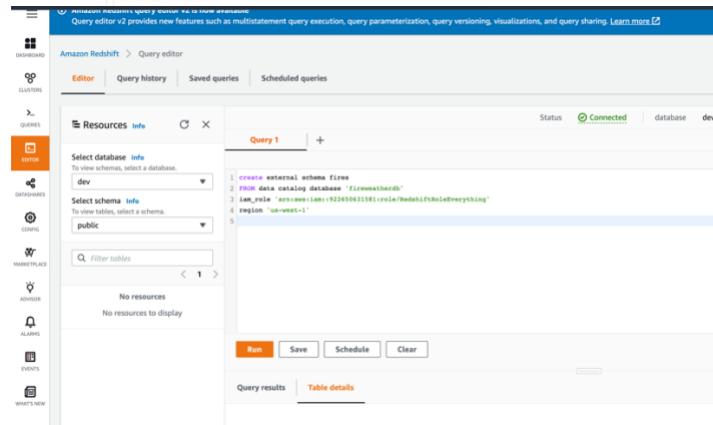
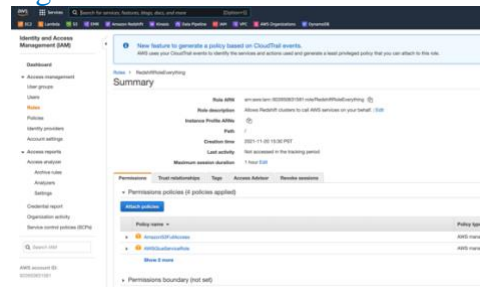
Run the following query with glue database name 'fireweatherdb' and the IAM role arn 'arn:aws:iam::922650631581:role/RedshiftRoleEverything'

create external schema fires

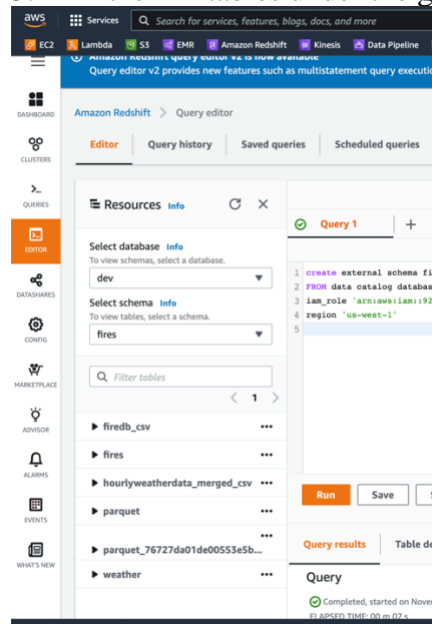
FROM data catalog database 'fireweatherdb'

iam\_role 'arn:aws:iam::922650631581:role/RedshiftRoleEverything'

region 'us-west-1'

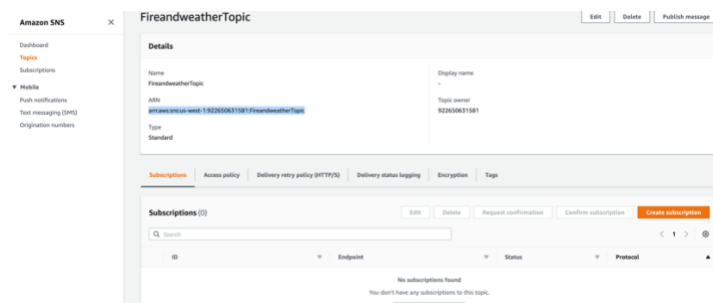
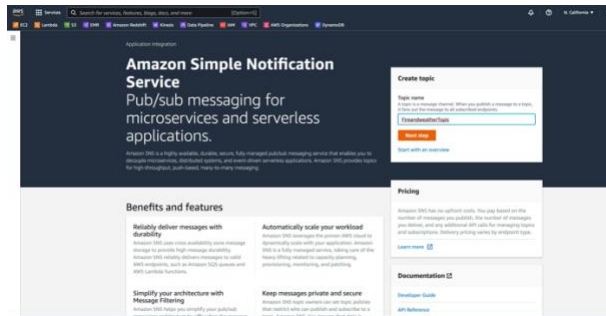


5. All the DB tables under the glue database will be accessible in the amazon cluster.

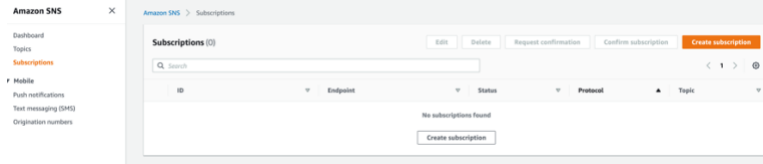


#####

Creating Lambda to send a mail once the tables are created  
Go to AWS SNS and create a topic of type “standard”



Create Subscription



Select the topic arn and enter a email ID

