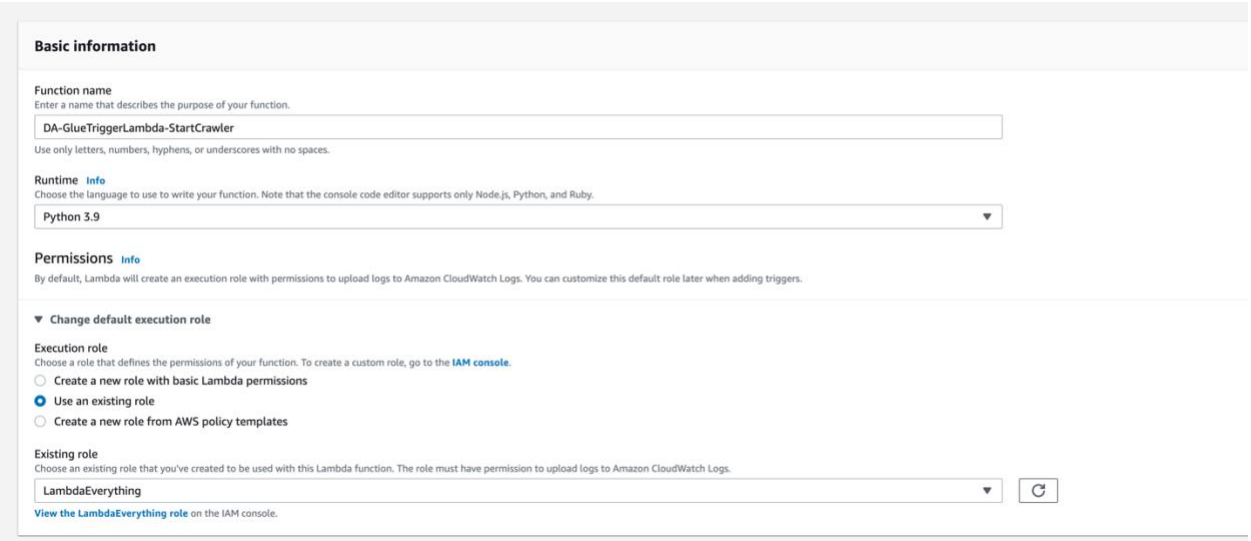
Table Of Contents

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



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

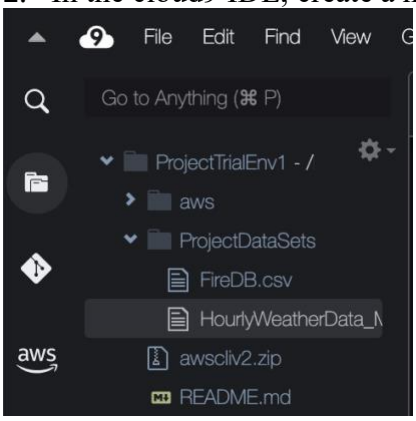
```
import json
import boto3
client=boto3.client('glue')
def lambda_handler(event, context):
    response = client.start_crawler(Name='FireAndWeatherCrawler')
    print(json.dumps(response, indent=4))
```

Creating Cloud 9 Environment

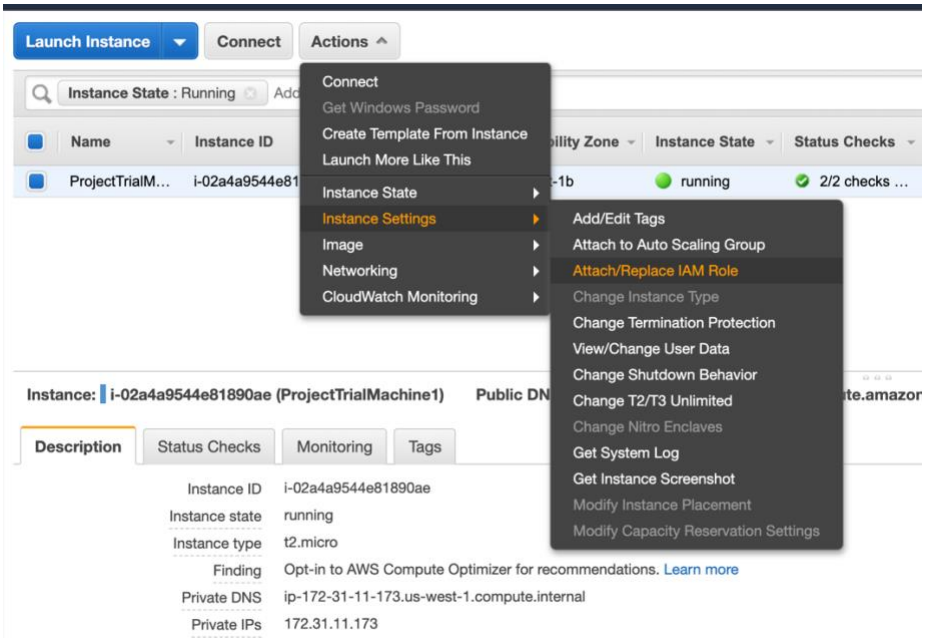
1. create a cloud9 environment with ec2 instance with all default configurations

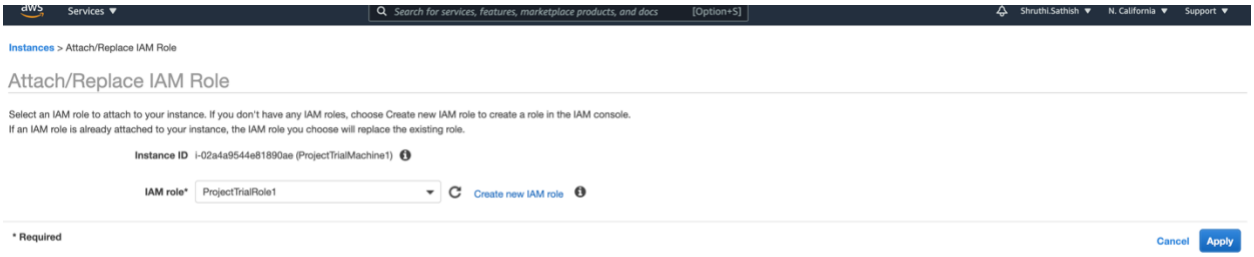
Go to EC2 instances and change name to “DataAvengersEC2”

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



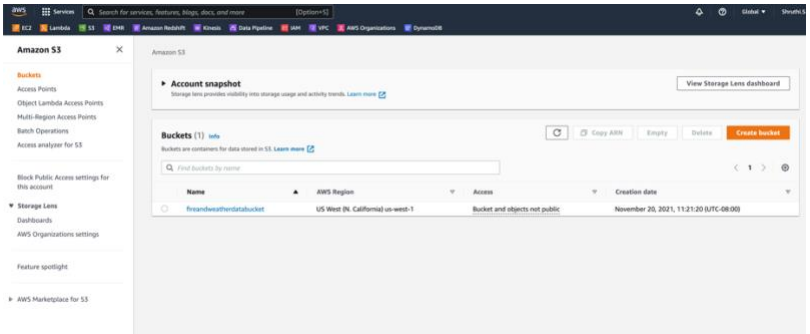
3. Create IAM Role to Allow EC2 instances to call AWS services on your behalf with full access to AmazonS3FullAccess, AWSLambda_FullAccess and AWSGlueServiceRole
4. Attach this AWS role to the ec2 instance



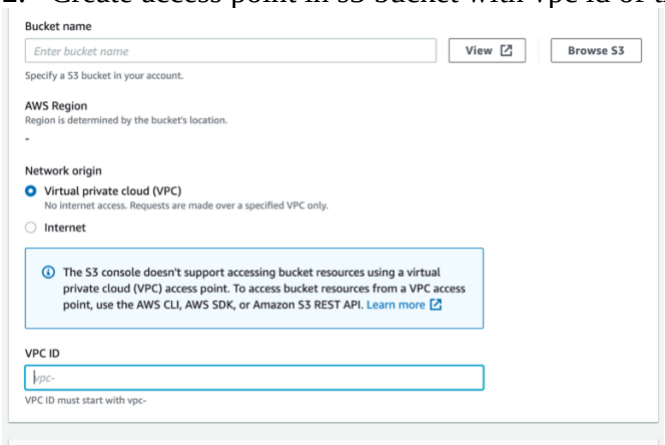


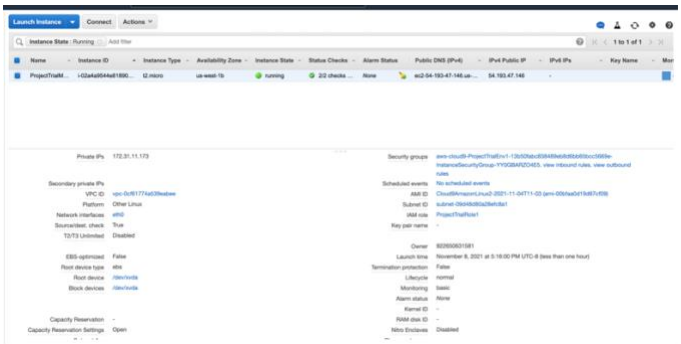
Create S3 Bucket

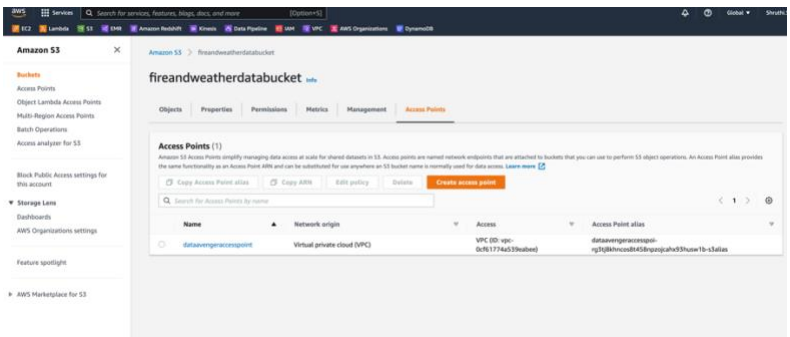
1. Create s3 bucket



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

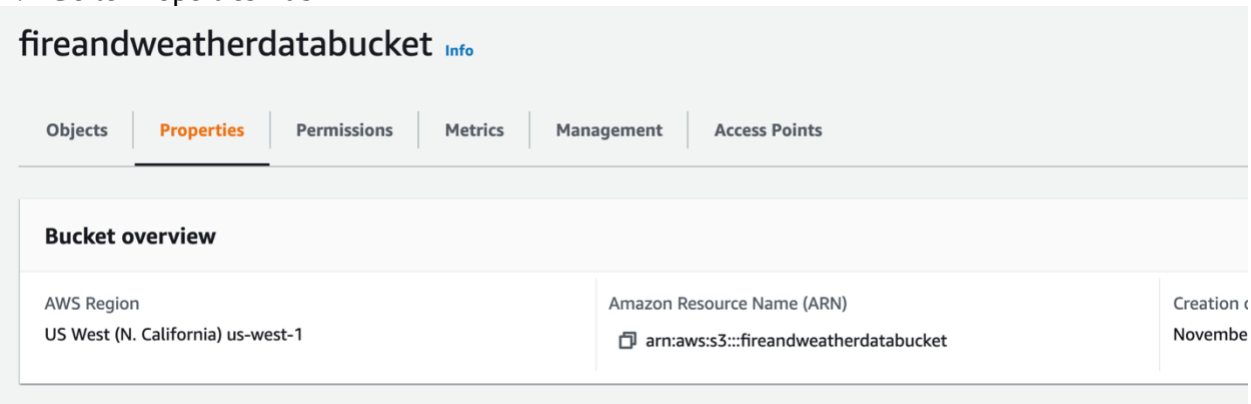




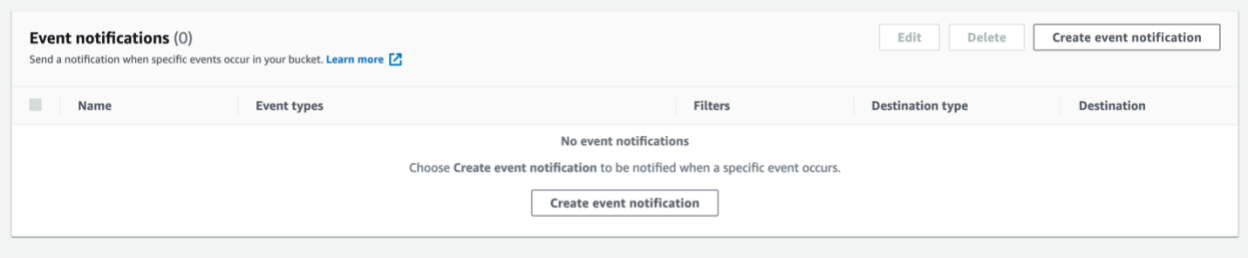


Adding Trigger To S3 Bucket

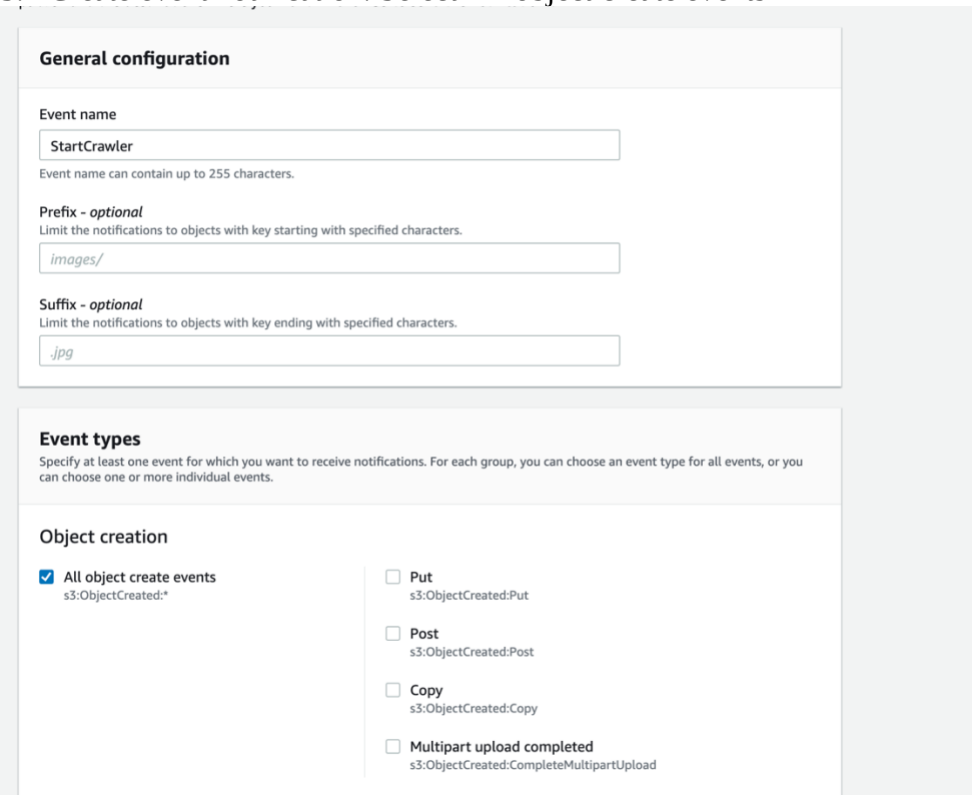
1. Go to Properties Tab



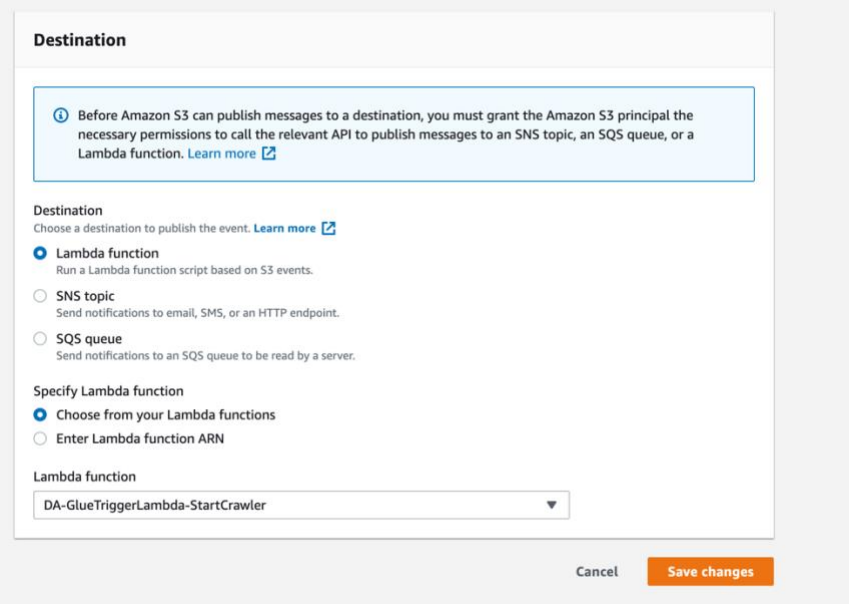
2. Scroll down to Event Notification



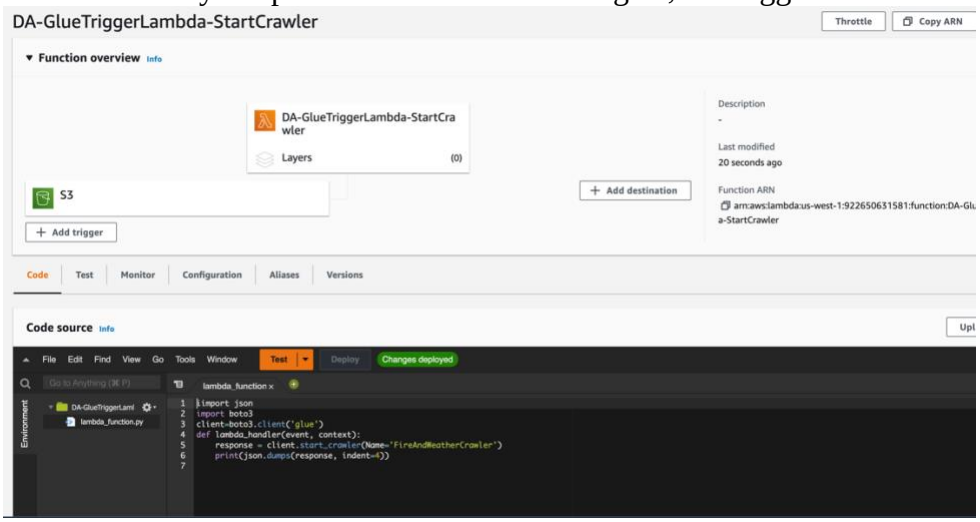
3. Create event notification. Select All object create events



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

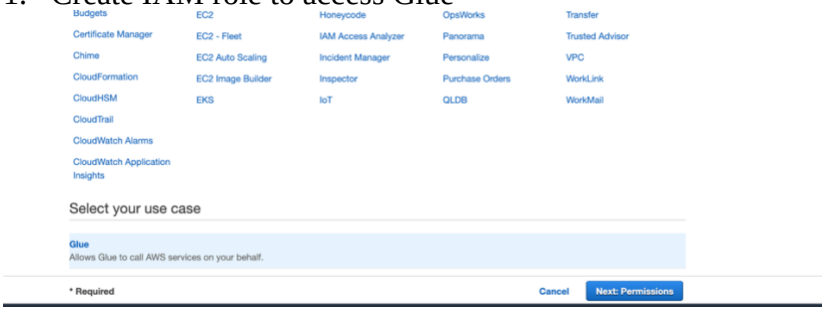


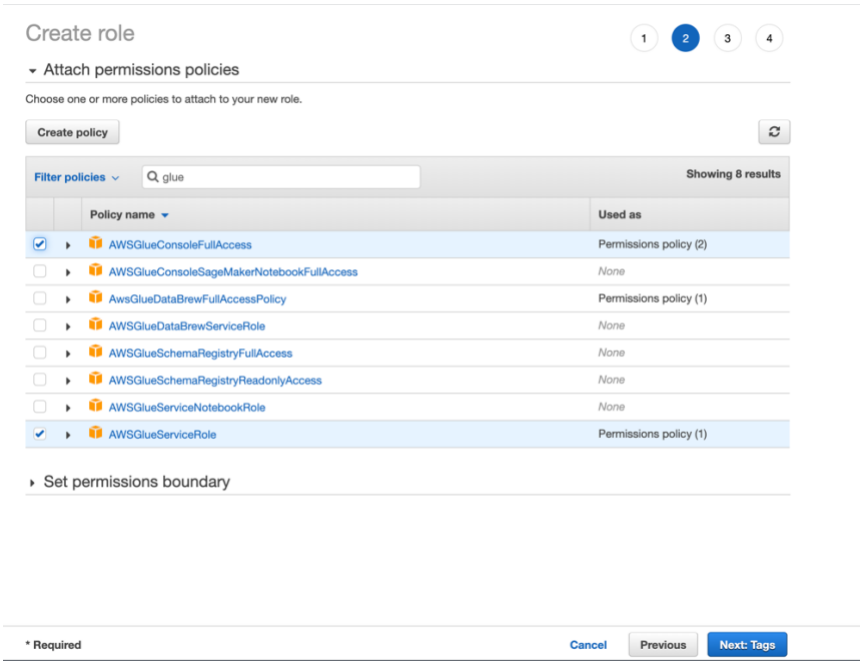
5. Now when you open the Lambda function again, the trigger will be added

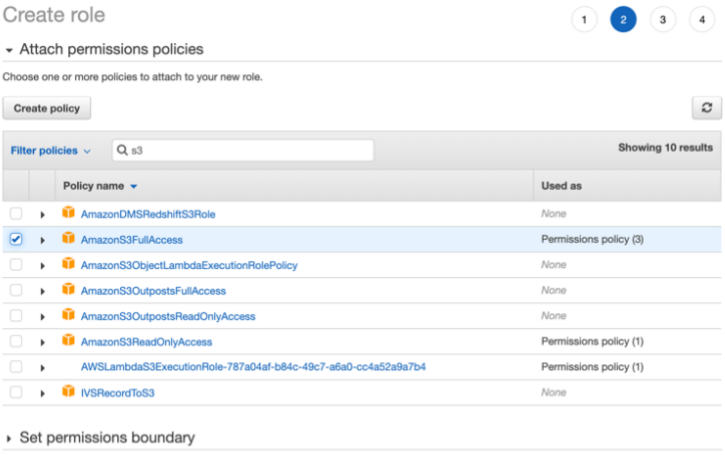


Creating The Crawler

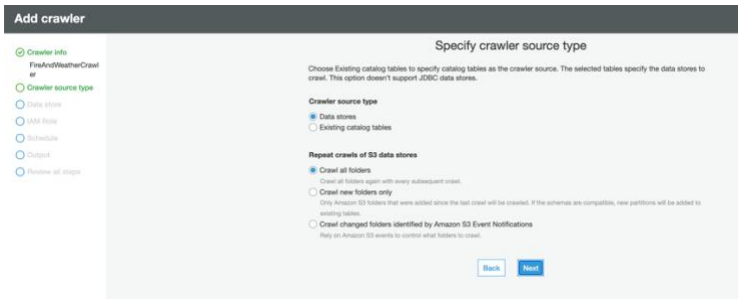
1. Create IAM role to access Glue



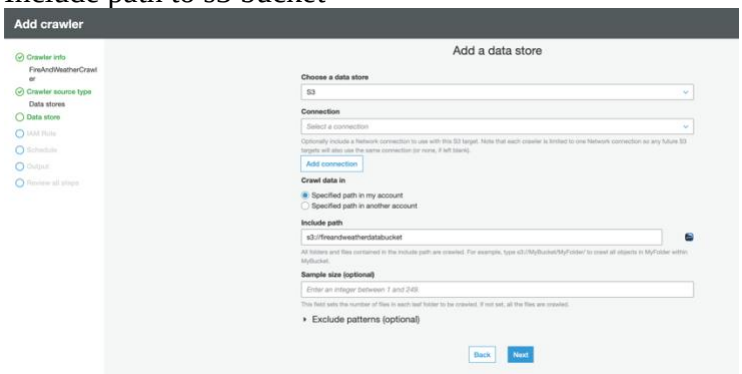


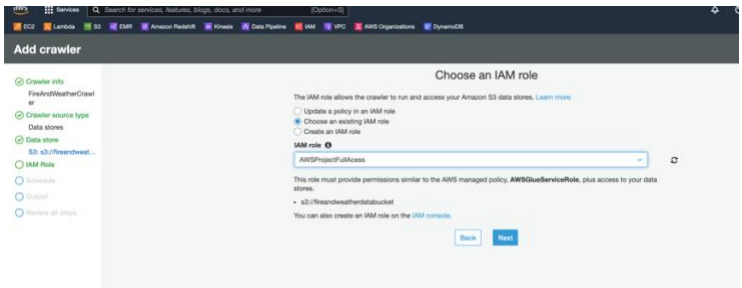


2. Create Glue crawler with IAM and create a new DB to create tables after the data is crawled
AWS Glue -> Crawlers -> Add Crawler

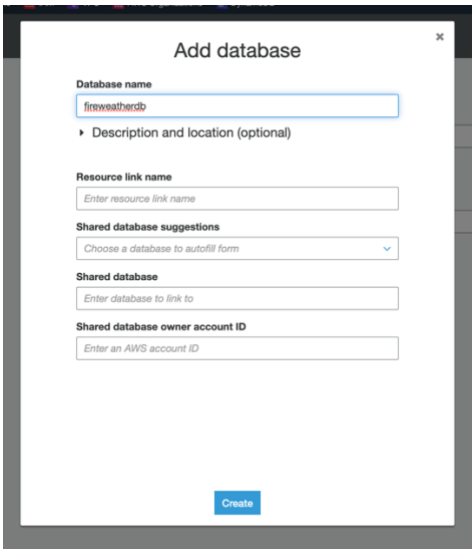


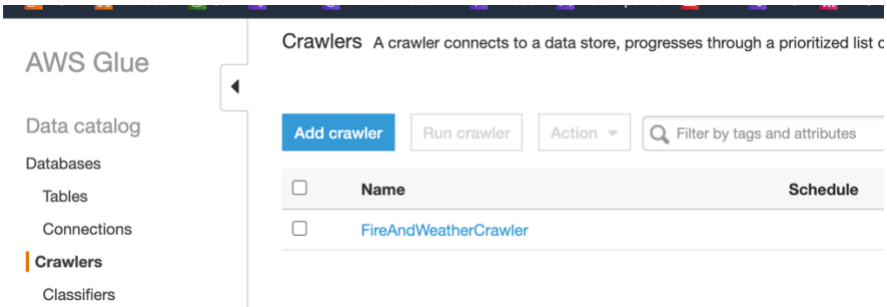
Include path to s3 bucket



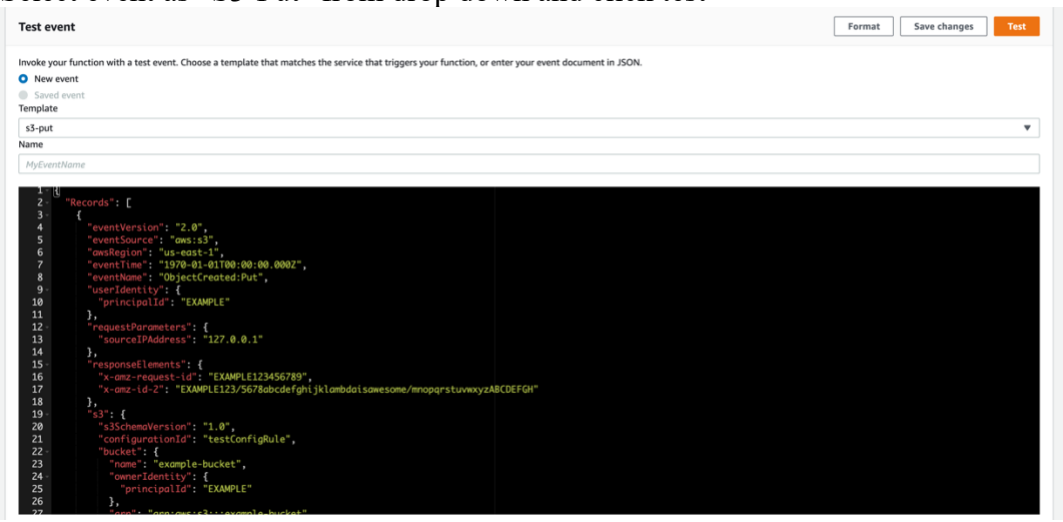


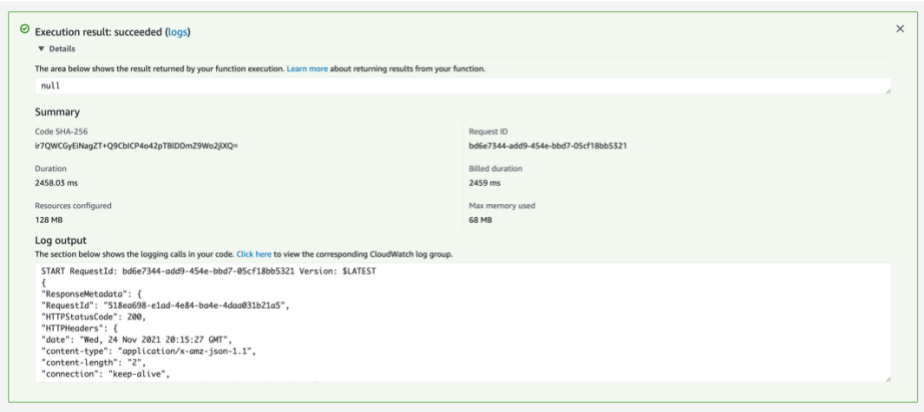
3. Add new Database





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

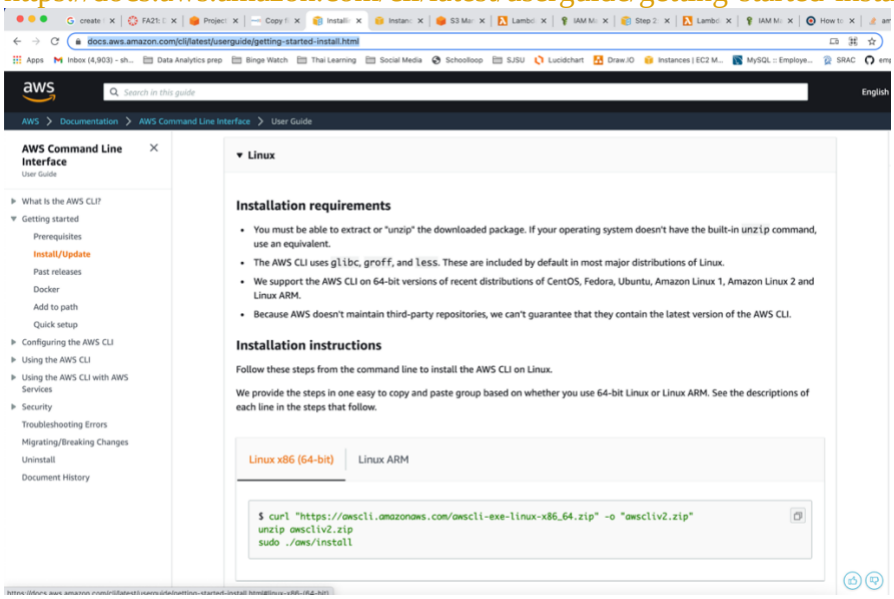




5. This means that the trigger is working properly

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 python36 get-pip.py                    # Install pip for Python 3.6.
sudo: python36: 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)

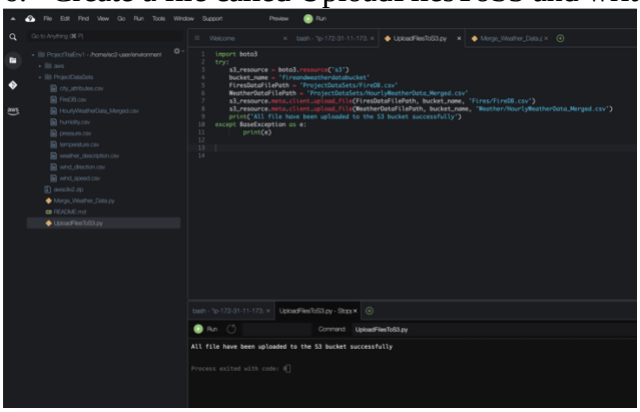
```
ec2-user:~/environment $ sudo python3 -m pip install boto3
WARNING: Running pip install with root privileges is generally not a good idea. Try `python3 -m pip install --user` instead.
Collecting boto3
  Downloading boto3-1.20.10-py3-none-any.whl (131 kB)
     |████████████████████████████████| 131 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

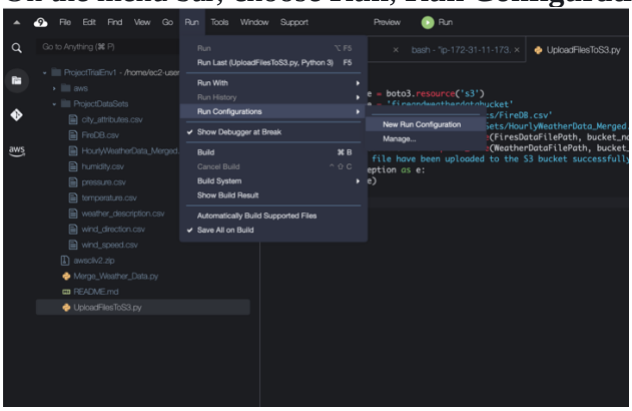
```
Complete!
ec2-user:~/environment $ sudo pip3 install pandas
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting pandas
  Downloading pandas-1.3.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.3 MB)
     |████████████████████████████████| 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-cp37-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 $
```

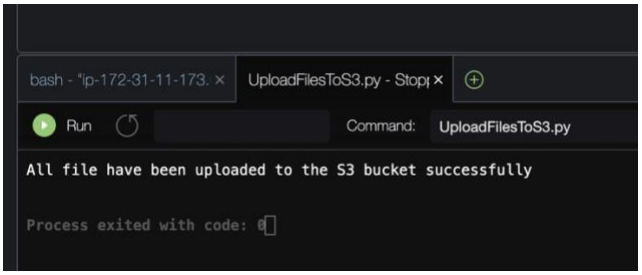
6. Create a file called UploadFilesToS3 and write the following code



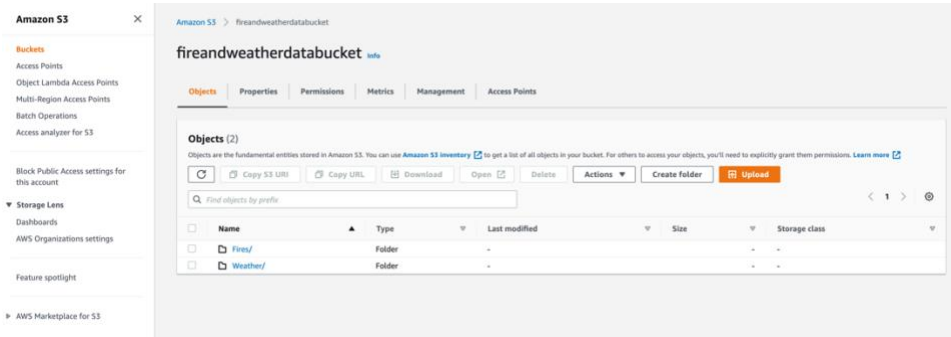
7. Execute the python script

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

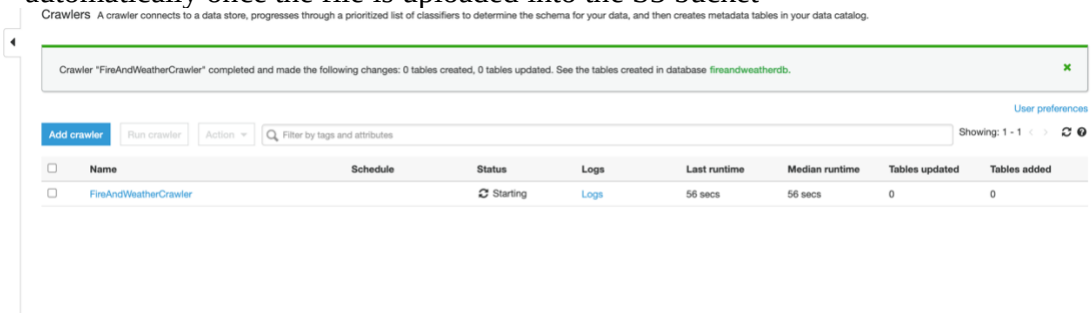




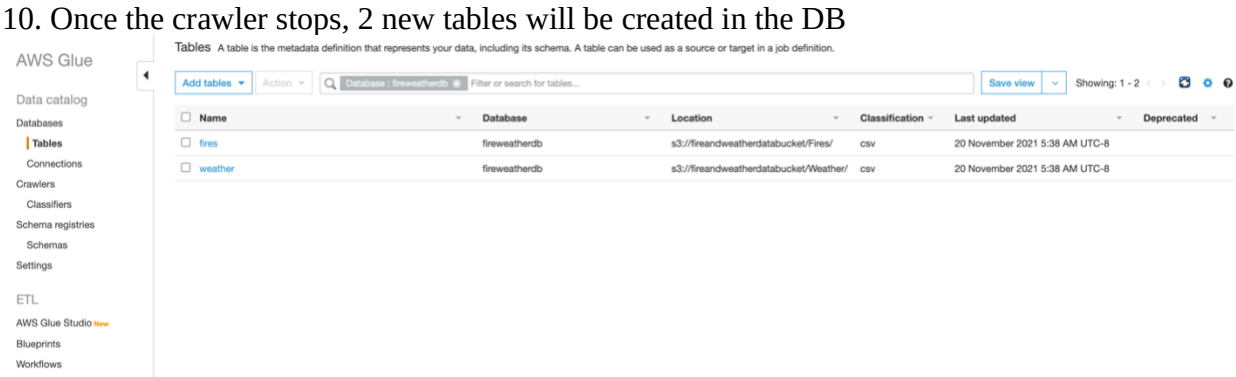
8. The data has been added to S3 Bucket



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

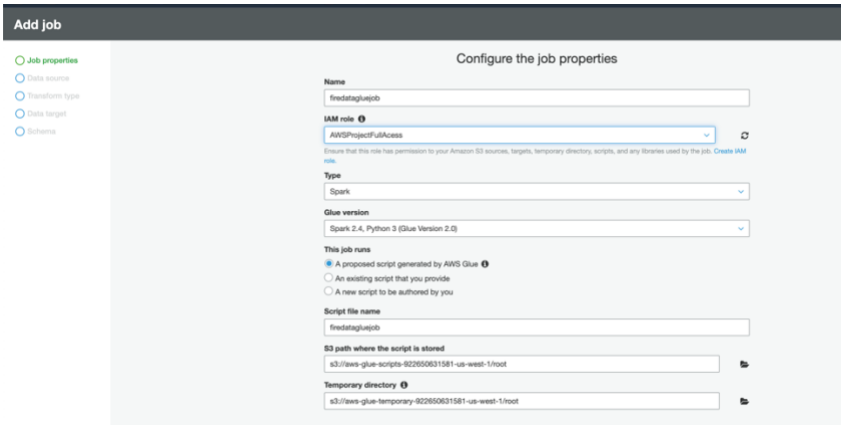


10. Once the crawler stops, 2 new tables will be created in the DB

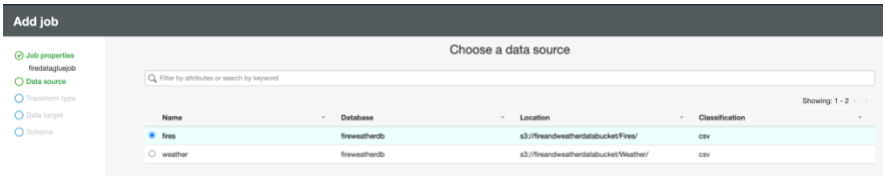


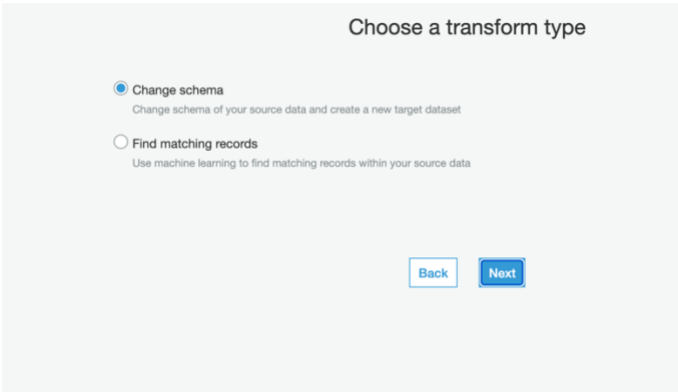
Aws Glue Jobs

1. Create Glue job with following Configurations
   AWS Glue -> ETL ->Jobs -> Add Job

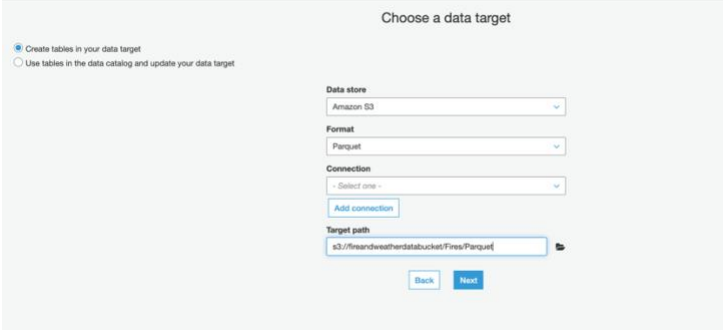


2. Select Fires as the datasource

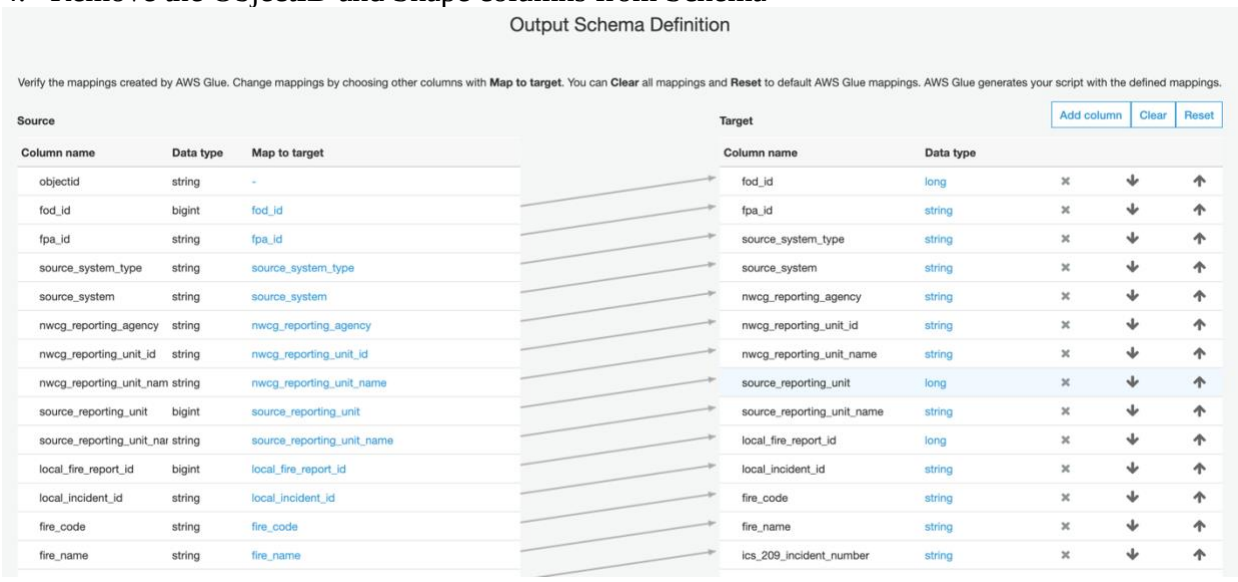




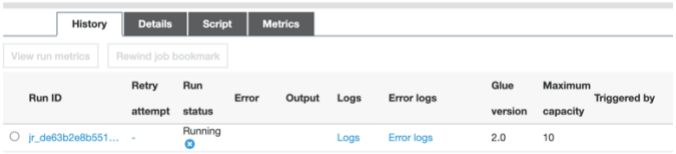
3. Choose target as S3, format Parquet and Target Path as /Fires/Parquet



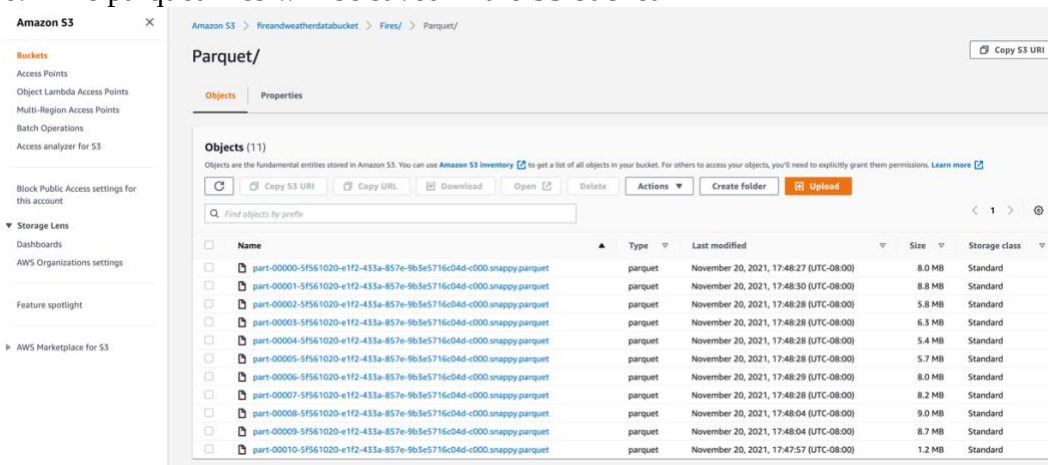
4. Remove the ObjectID and Shape columns from Schema



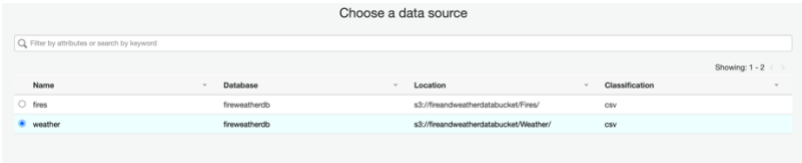
5. Save the script and run the job

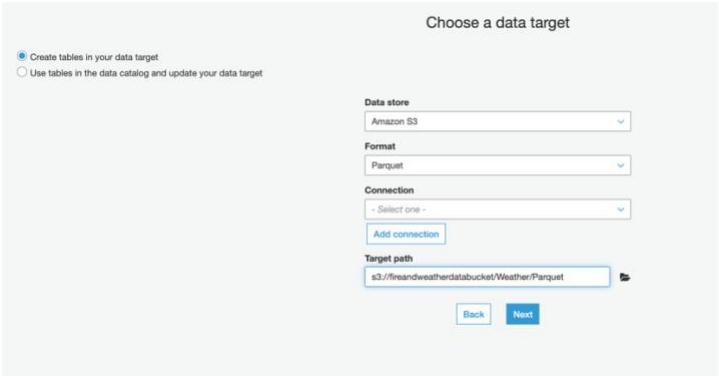


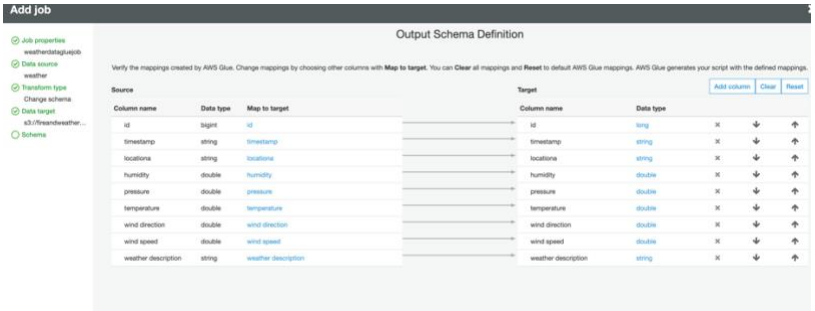
6. The parquet files will be saved in the S3 bucket



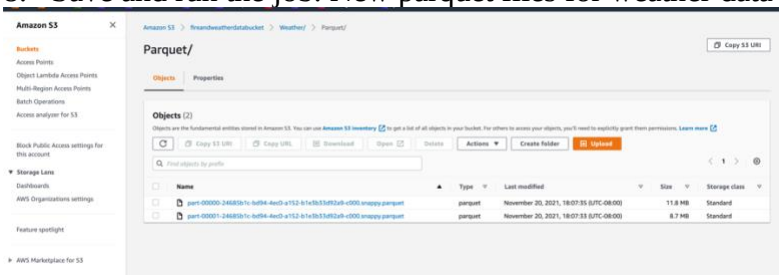
7. Create another Glue job for weather data







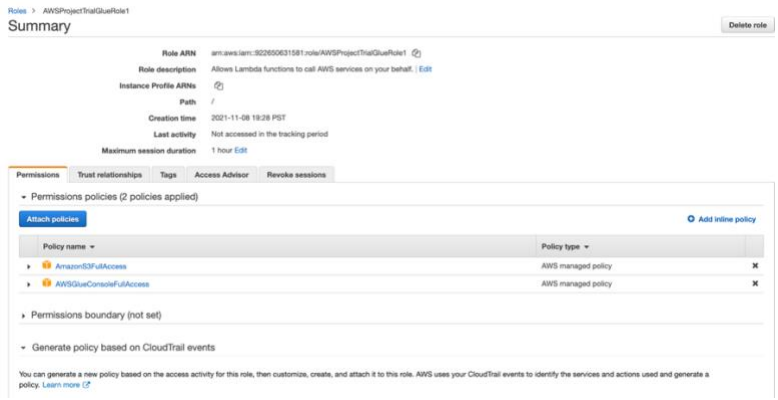
8. Save and run the job. New parquet files for weather data will be added to S3 bucket



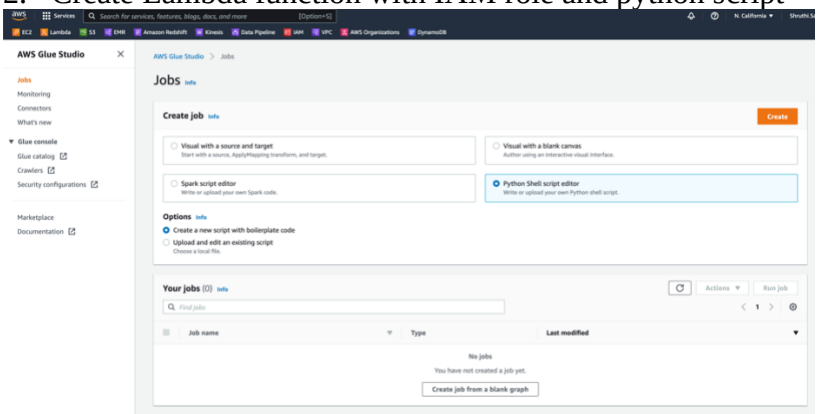
Adding Lambdas To Invoke Glue Jobs From Crawler

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



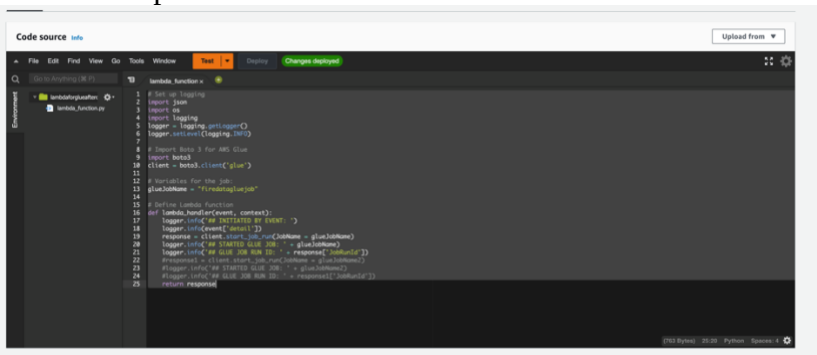
2. Create Lambda function with IAM role and python script



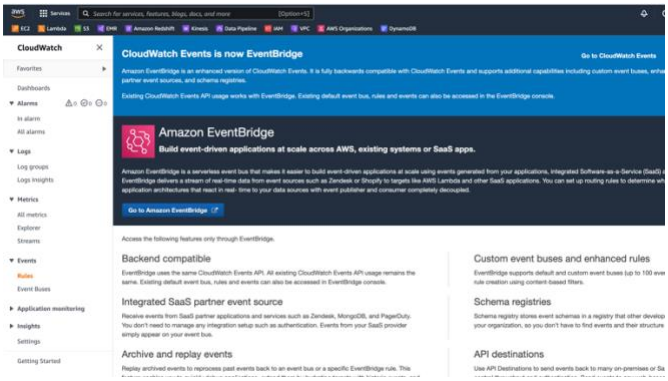
Add the code

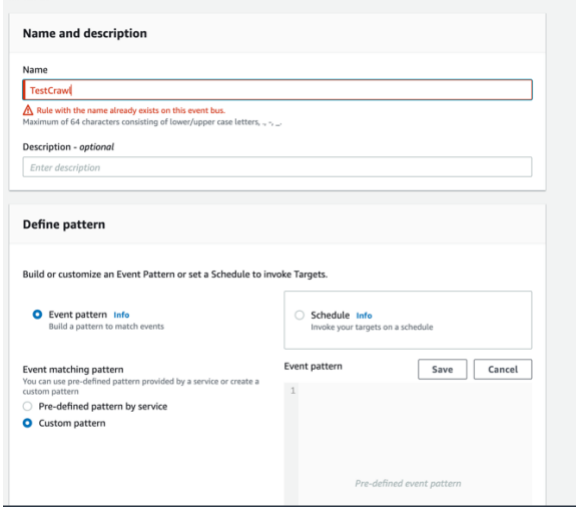
```
# Set up logging
import json
import os
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# 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
```



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



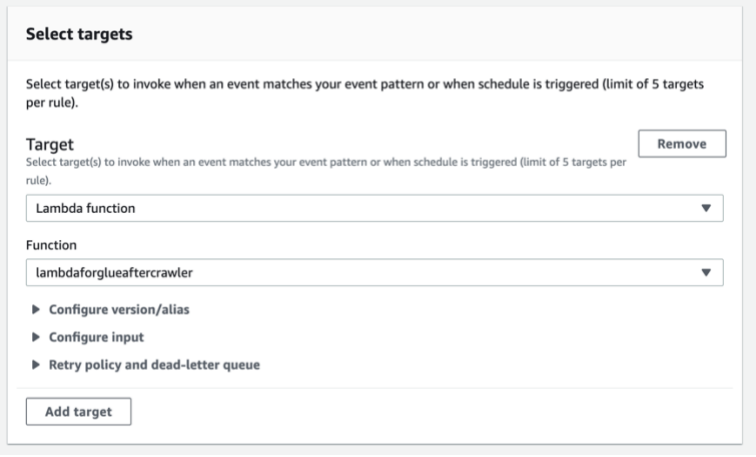


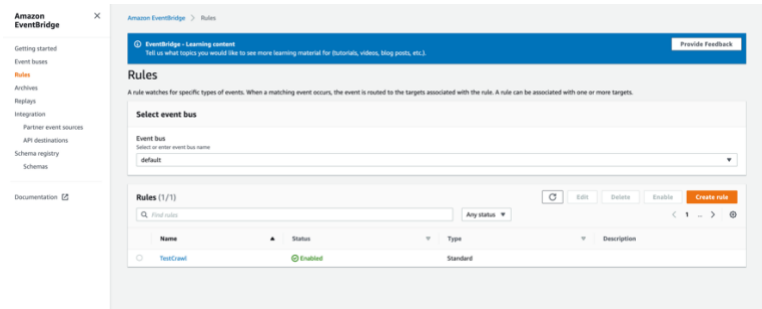
In Custom pattern add the code and change the crawler name to our crawler

```
{
    "detail-type": [
```

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

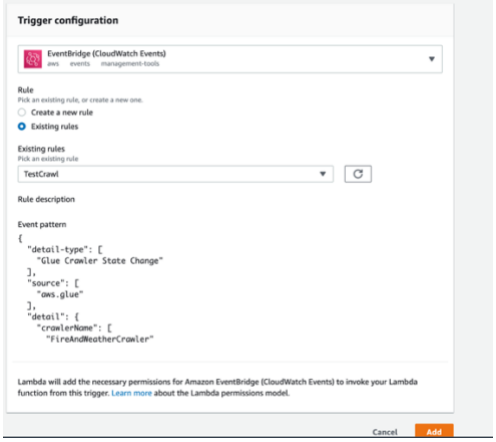
4. Add the lambda function in target and click create





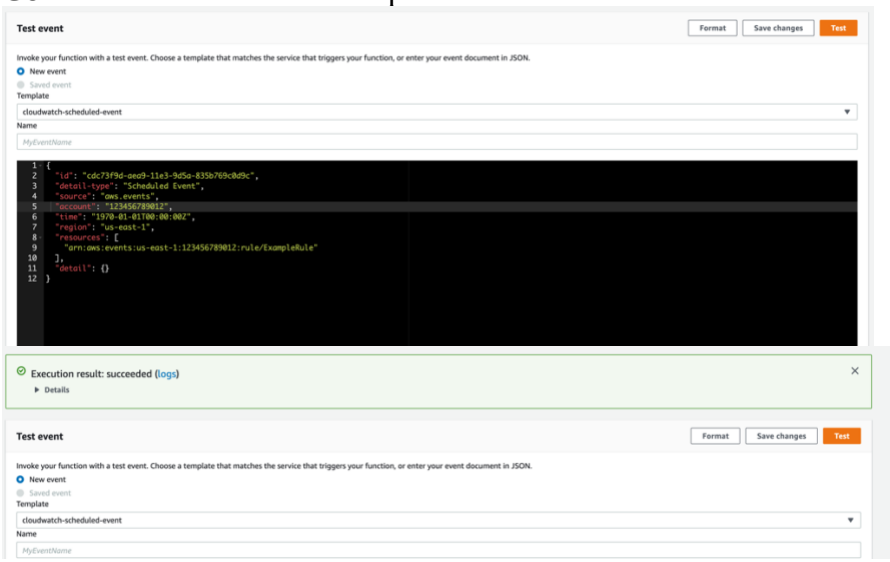
5. Go back to Lambda Function-> Add trigger

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

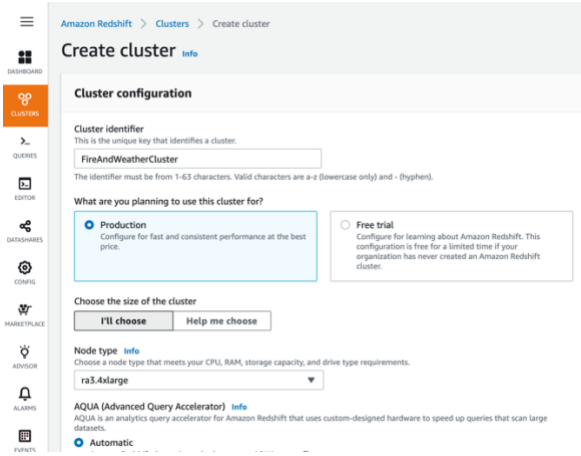


7. Repeat the same and create another lambda function for “WeatherGlueJob”. But we can skip the rule creation. We can use the same rule we created for previous function. Everything else remains the same.

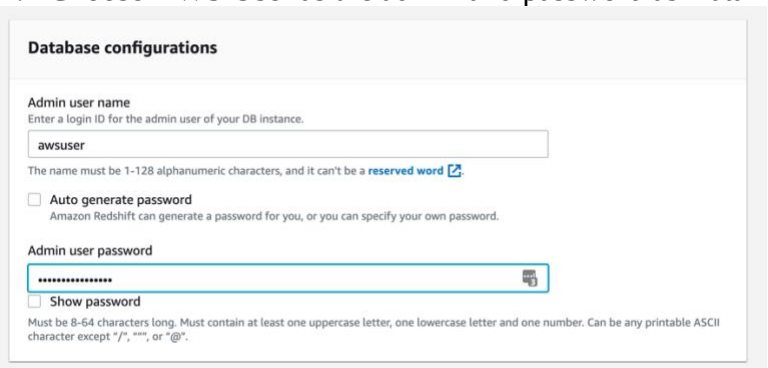
Now, go back to Glue and run the crawler. Once it stops running, go to jobs and check the history to check if the glue jobs are triggered.

Create Red Shift Cluster

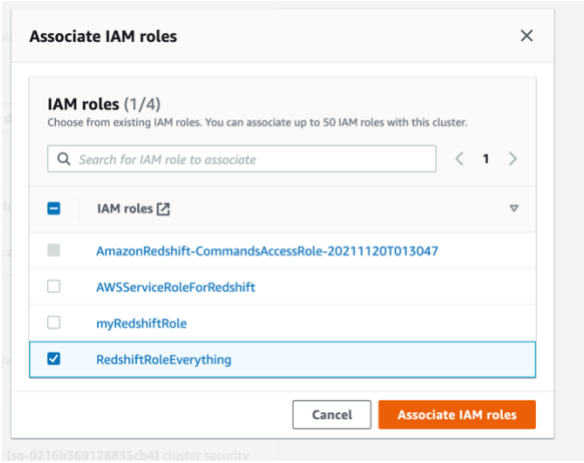
1. Create a RedShift Cluster



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



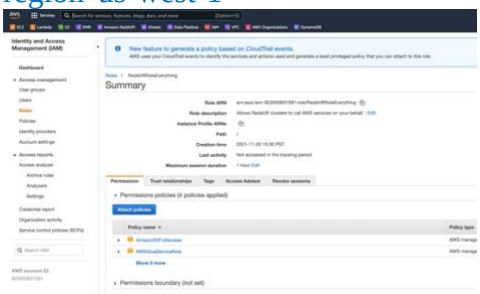
3. In IAM roles, choose the one with full access to s3, glue and redshift and create cluster

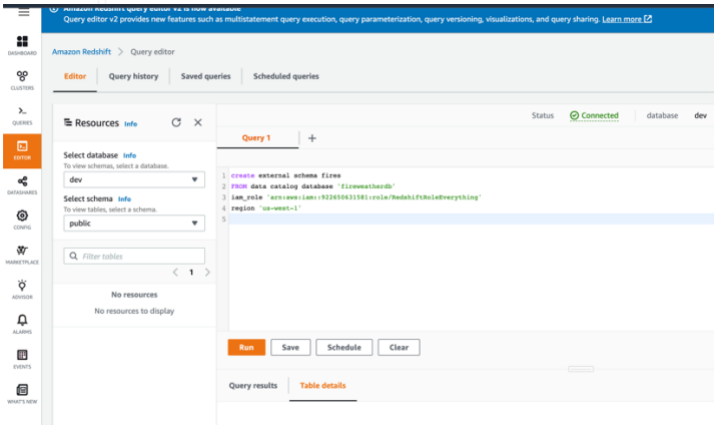


4. In the Query Editor of amazon cluster. Click on connect to database and it will automatically connect to dev database as default

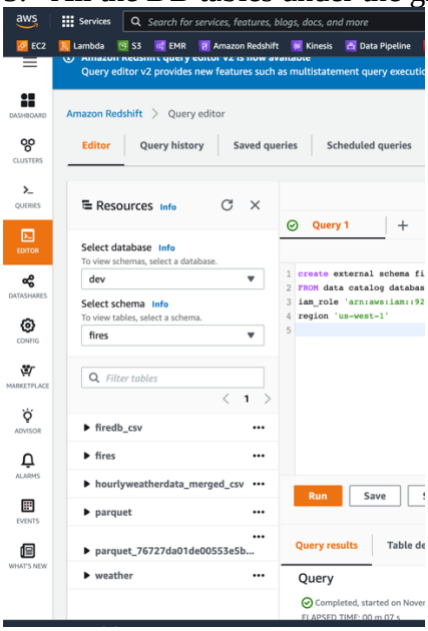
Run the following query with glue database name ‘fireweatherdb’ and the IAM role arn 'arn:aws:iam::922650631581:role/RedshiftRoleEverything'

```
create external schema fires
FROM data catalog database 'fireweatherdb'
iam_role 'arn:aws:iam::922650631581:role/RedshiftRoleEverything'
region 'us-west-1'
```

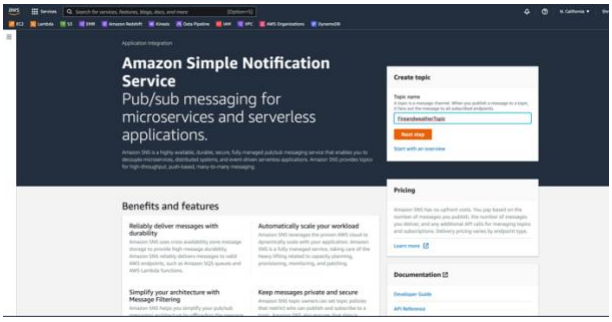


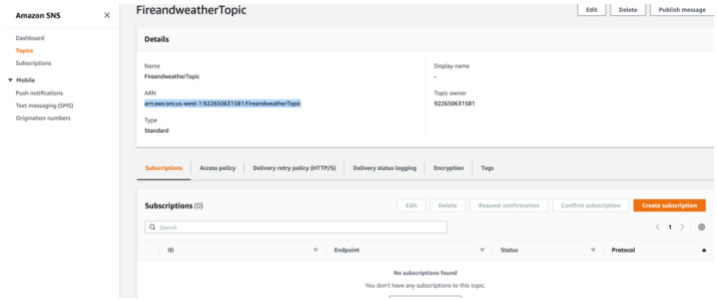


5. All the DB tables under the glue database will be accessible in the amazon cluster.

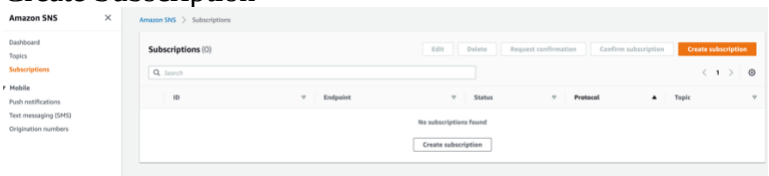


##
Creating Lambda to send a mail once the tables are created
Go to AWS SNS and create a topic of type “standard”





Create Subscription



Select the topic arn and enter a email ID