[Version 1.0.21]

Performing ETL on a Dataset by Using AWS Glue

Lab overview and objectives

Big data problems often involve a large number of heterogeneous data sources. As a data analyst, you might not know the schema for some data sources. This is the *variety* aspect of the five Vs of big data (volume, variety, velocity, veracity, and value). In this lab, you will work with AWS Glue to perform extract, transform, and load (ETL) for a dataset. You can direct AWS Glue to a data source, and it can infer a schema based on the data types that it discovers. Then, AWS Glue builds a Data Catalog that contains metadata about the various data sources.

AWS Glue is similar to Amazon Athena in that the actual data that you analyze remains in the data source. The key difference is that you can build a crawler with AWS Glue to discover the schema and then extract the data from the dataset. You can also transform the schema and then load the data into an AWS Glue database. You can then analyze the data by using SQL statements in Athena.

In this lab, you will learn how to use AWS Glue to import a dataset from Amazon Simple Storage Service (Amazon S3). You will then extract the data, transform its schema, and load the dataset into an AWS Glue database for later analysis by using Athena.

After completing this lab, you will be able to do the following:

- Access AWS Glue in the AWS Management Console and create a crawler.
- Create an AWS Glue database with tables and a schema by using a crawler.
- Query data in the AWS Glue database by using Athena.
- Create and deploy an AWS Glue crawler by using an AWS CloudFormation template.
- Review an AWS Identity and Access Management (IAM) policy for users to run an AWS Glue crawler and query an AWS Glue database in Athena.
- Confirm that a user with the IAM policy can use the AWS Command Line Interface (AWS CLI) to access the AWS Glue database that the crawler created.
- Confirm that a user can run the AWS Glue crawler when source data changes.

Duration

This lab will require approximately **90 minutes** to complete.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

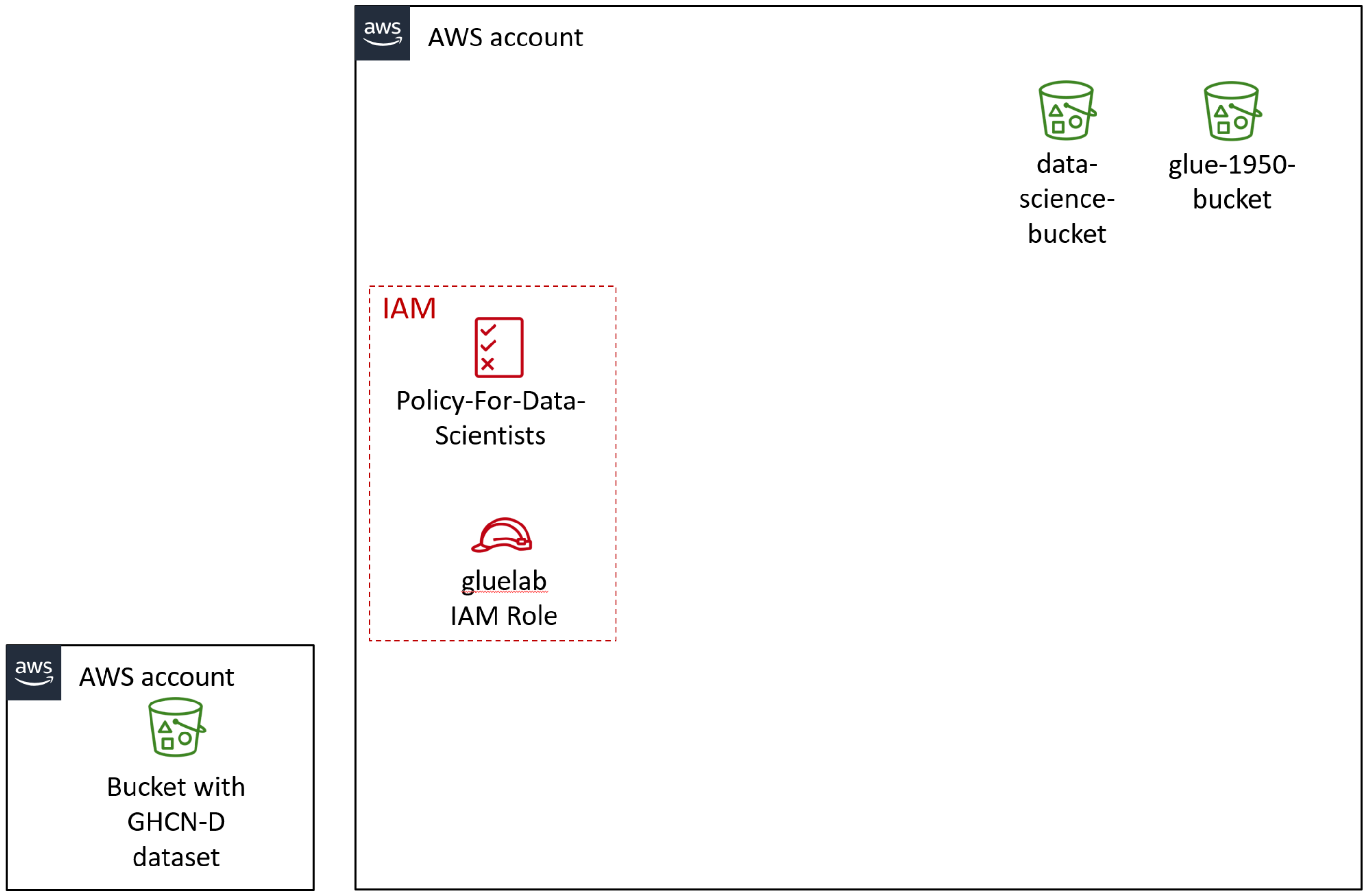
The data science team has asked you to create a series of proofs of concept (POCs) to use AWS services to address many of the data engineering and data analysis needs of the university. Mary, one of the data science team members, has seen what Athena can do to create tables that have defined schemas and is impressed. She asks you if it's possible to infer the columns and data types automatically. Defining the schema takes much of her time when she deals with large amounts of varied data. You want to develop a POC to use AWS Glue, which is designed for use cases that are similar to this one.

To develop a POC, Mary suggests that you use a publicly available dataset, the Global Historical Climatology Network Daily [GHCN-D] dataset, contains daily weather summaries from ground-based stations, going back to 1763. The dataset is publicly available in an S3 bucket.

Mary explains that the most common recorded parameters in the dataset are daily temperatures, rainfall, and snowfall. These parameters are useful to assess risks for drought, flooding, and extreme weather. The data definitions used in this lab are available on the NOAA Global Historical Climatology Network Daily (GHCN-D) Dataset page.

Note: As of October 2022, the dataset has been split into sub-datasets, **by_year** and **by_station**. Throughout this lab you will be using **by_year** and it can be found at **s3://noaa-ghcn-pds/csv/by_year/**.

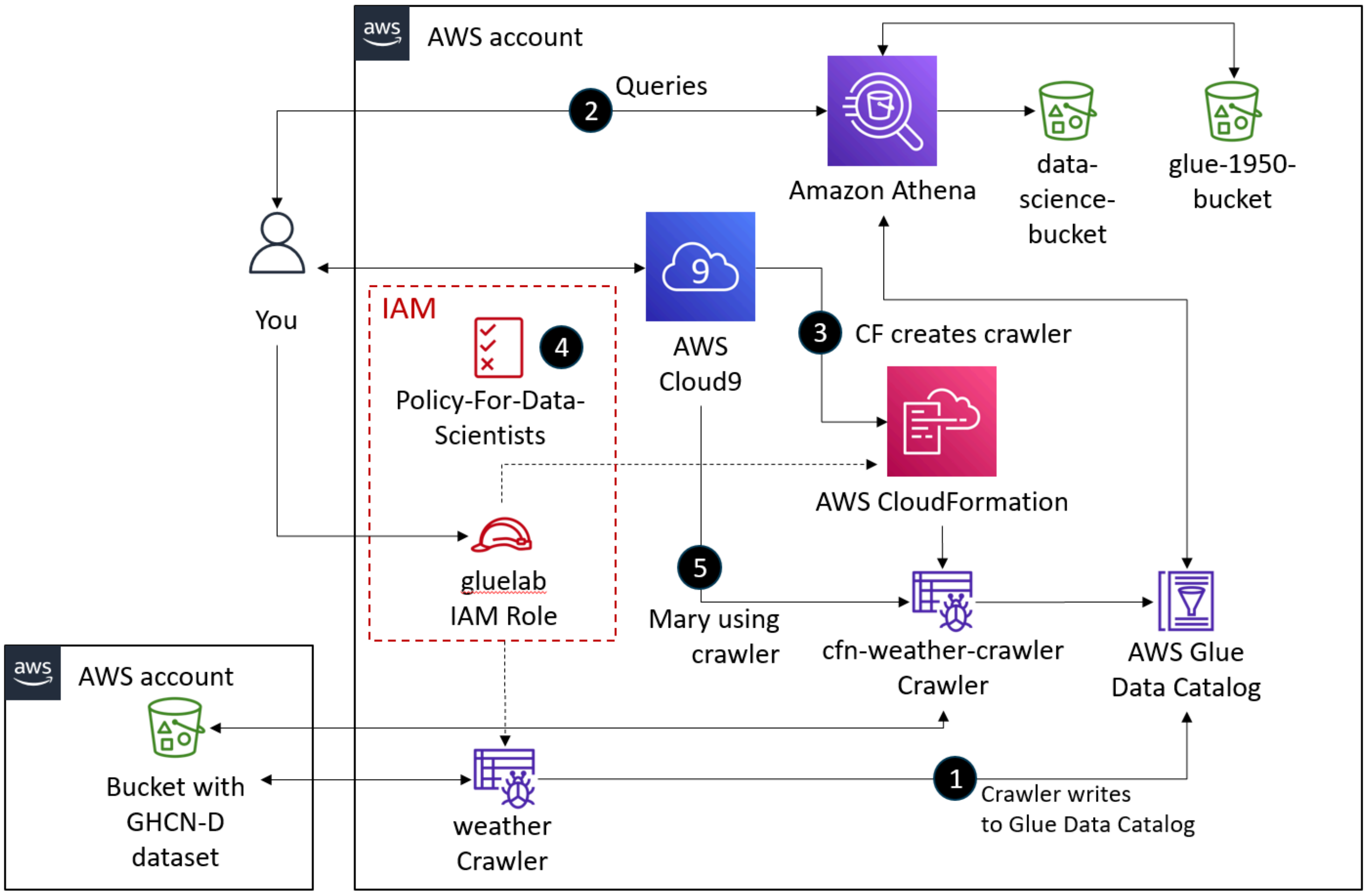
When you *start* the lab, the environment will contain the resources that are shown in the following diagram. For this lab environment, the original data source is an S3 bucket that exists in another AWS account.



The lab environment is created with a CloudFormation template that is deployed when you launch the lab. The resulting CloudFormation stack creates two S3 buckets entitled *data-science-bucket* and *glue-1950-bucket*, an IAM policy entitled *Policy-For-Data-Scientists*, and an IAM role entitled *gluelab*.

Tip: To review the CloudFormation template that built this environment, navigate to the CloudFormation console. In the navigation pane, choose **Stacks**.

By the *end* of the lab, you will have created the additional architecture shown in the following diagram. The table after the diagram provides a detailed explanation of the architecture and how it relates to the tasks that you will complete in this lab.



Numbered Task	Detail
1	You will create and use an AWS Glue crawler named *weather* with the *gluelab* IAM bucket in another AWS account. The crawler will populate the *weatherdata* databas
2	You will also use the Glue console to transform the database by modifying its scher
3	When the Data Catalog is ready, you will use Athena to query the database and bui stored in the *data-science-bucket* S3 bucket.
4	You will create an AWS Glue database table using another query that only includes in the *glue-1950-bucket* S3 bucket.
5	You will create views and use these to calculate the the average temperature of eac
6	You will use the AWS CLI within the AWS Cloud9 terminal to create a CloudFormati *crawler-weather*. Other team members and other university departments will be abl
7	You will review the *Policy-For-Data-Scientists* IAM policy to understand the team's a
8	You will test Mary's access to the *cfn-crawler-weather* crawler and run it by using he

Accessing the AWS Management Console

1. At the top of these instructions, choose **Start Lab**.
   - The lab session starts.
   - A timer displays at the top of the page and shows the time remaining in the session.

     Tip: To refresh the session length at any time, choose **Start Lab** again before the timer reaches 00:00.
   - Before you continue, wait until the circle icon to the right of the AWS link in the upper-left corner turns green.
2. To connect to the AWS Management Console, choose the **AWS** link in the upper-left corner.
   - A new browser tab opens and connects you to the console.

     Tip: If a new browser tab does not open, a banner or icon is usually at the top of your browser with the message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose **Allow pop-ups**.

Task 1: Using an AWS Glue crawler with the GHCN-D dataset

As a data engineer or analyst, you might not always know the schema of the data that you need to analyze. AWS Glue is designed for this situation. You can direct AWS Glue to data that is stored on AWS, and the service will discover your data. AWS Glue will then store the associated metadata (for example, the table definition and schema) in the AWS Glue Data Catalog. You accomplish this by creating a crawler, which inspects the data source and infers a schema based on the data.

In this task, you will work with data that is publicly available in an S3 bucket to do the following:

- Configure and create an AWS Glue crawler.
- Run the crawler to extract, transform, and load data into an AWS Glue database.
- Review the metadata of a table that the crawler created.
- Edit the schema of the table.

First, you will configure and create a crawler to discover the schema for the GHCN-D dataset and extract the data from it.

3. Configure and create the AWS Glue crawler.
   - In the AWS Management Console, in the search box next to **Services**, search for and choose **AWS Glue** to open the AWS Glue console.
   - In the navigation pane, under **Databases**, choose **Tables**.
   - Choose **Add tables using crawler**.
   - For **Name**, enter `Weather`
   - Expand the **Tags (optional)** section.

Notice that this is where you could add tags or extra security configurations. Keep the default settings.

- Choose **Next** at the bottom of the page.
- Choose **Add a data source** and configure the following:
  - **Data source:** Choose **S3**.
  - **Location of S3 data:** Choose **In a different account**.
  - **S3 path:** Enter the following S3 bucket location for the publicly available dataset:

    ```
    s3://noaa-ghcn-pds/csv/by_year/
    ```

  - **Subsequent crawler runs:** Choose **Crawl all sub-folders**.
  - Choose **Add an S3 data source**.
- Choose **Next**.
- For **Existing IAM role**, choose **gluelab**.

  This role was provided in the lab environment for you. For reference, see the lab's CloudFormation template. The following is the YAML snippet for this role:

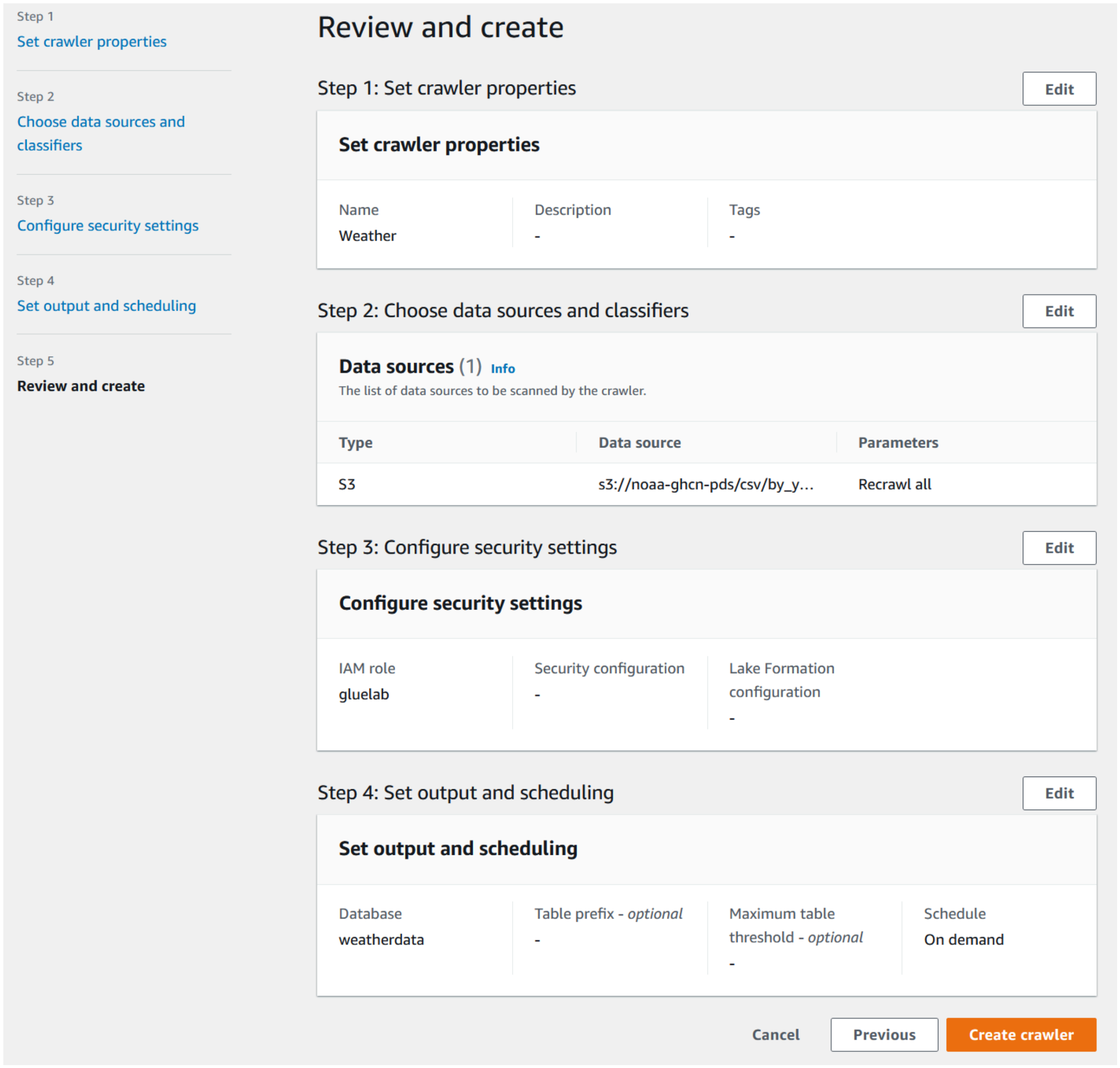
  ```
  GlueLab:
      Type: AWS::IAM::Role
      Properties:
        RoleName: "gluelab"
        Path: "/"
        AssumeRolePolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Principal:
                Service:
                  - glue.amazonaws.com
              Action:
                - sts:AssumeRole
        ManagedPolicyArns:
          - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
          - arn:aws:iam::aws:policy/AmazonS3FullAccess
  ```

- Choose **Next**.
- In the **Output configuration** section, choose **Add database**.

  A new browser tab opens.
- For **Name**, enter `weatherdata`
- Choose **Create database**.

- Return to the browser tab that is open to the **Set output and scheduling** page in the AWS Glue console.
- For **Target database**, choose the **weatherdata** database that you just created.

  Tip: To refresh the list of available databases, choose the refresh icon to the right of the dropdown list.
- In the **Crawler schedule** section, for **Frequency**, keep the default **On demand**.
- Choose **Next**.
- Confirm your crawler configuration is similar to the following.


Step 1
Set crawler properties
Step 2
Choose data sources and classifiers
Step 3
Configure security settings
Step 4
Set output and scheduling
Step 5
Review and create

Review and create

Step 1: Set crawler properties

Set crawler properties

Name	Description	Tags
Weather	-	-

Step 2: Choose data sources and classifiers

Data sources (1) Info
The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://noaa-ghcn-pds/csv/by_y...	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role	Security configuration	Lake Formation configuration
gluelab	-	-

Step 4: Set output and scheduling

Set output and scheduling

Database	Table prefix - *optional*	Maximum table threshold - *optional*	Schedule
weatherdata	-	-	On demand

- Choose **Create crawler**.

  To perform the extract and load steps of the ETL process, you will now run the crawler.

  You can create AWS Glue crawlers to either run on demand or on a set schedule. Because you created your crawler to run on demand, you must *run* the crawler to build the database and generate the metadata.

4. Run the crawler.
   - On the **Crawlers** page, select the **Weather** crawler that you just created.
   - Choose **Run**.

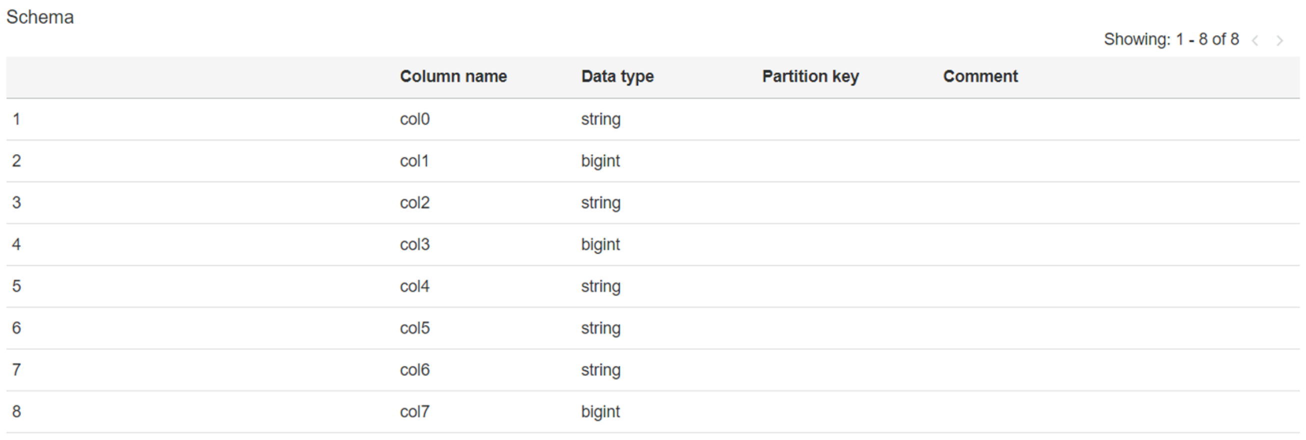
     The crawler state changes to *Running*.

     Important: Wait for the status to change to *Ready* before moving to the next step. This will take about 3 minutes.

     AWS Glue creates a table to store metadata about the GHCN-D dataset. Next, you will inspect the data that AWS Glue captured about the data source.

5. Review the metadata that AWS Glue created.
   - In the navigation pane, choose **Databases**.
   - Choose the link for the **weatherdata** database.
   - In the **Tables** section, choose the **by_year** link.

     Review the metadata that the weather crawler captured, as shown in the following screenshot. The schema lists the columns that the crawler discovered in the imported dataset.



Schema

Showing: 1 - 8 of 8

	Column name	Data type	Partition key	Comment
1	col0	string		
2	col1	bigint		
3	col2	string		
4	col3	bigint		
5	col4	string		
6	col5	string		
7	col6	string		
8	col7	bigint		

     Now you will edit the schema of the database, which is part of transforming data in the ETL process.

6. Edit the schema.
   - From the **Actions** menu in the upper-right corner of the page, choose **Edit schema**.
   - Change the column names according to the following table.

     To change a column name, select the check box for the item that you want to modify, and then choose **Edit**.

     In the window that opens, change the value for the **Name**, and then choose **Edit**. Repeat these steps for each column name.

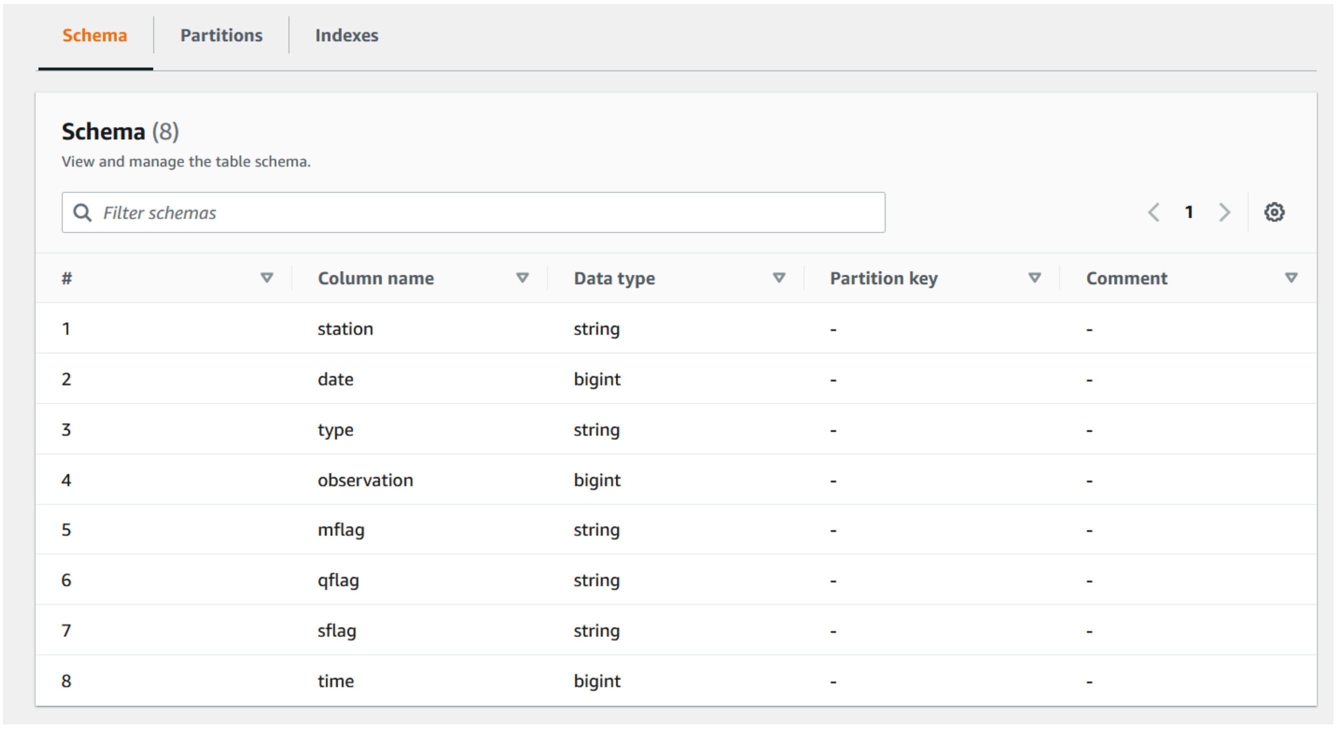
     Note: AWS Glue supports column names in lowercase only.

Previous Name	New Name
id	`station`

Previous Name	New Name
date	`date`
element	`type`
data_value	`observati`
m_flag	`mflag`
q_flag	`qflag`
s_flag	`sflag`
obs_time	`time`

- Choose **Update schema**.

  The schema for the table now looks like the following screenshot.



#	Column name	Data type	Partition key	Comment
1	station	string	-	-
2	date	bigint	-	-
3	type	string	-	-
4	observation	bigint	-	-
5	mflag	string	-	-
6	qflag	string	-	-
7	sflag	string	-	-
8	time	bigint	-	-

Task 1 summary

In this task, you used the console to create a crawler in AWS Glue. You directed the crawler to data that is stored in an S3 bucket, and the crawler discovered the data. Then, the crawler stored the associated metadata (the table definition and schema) in a Data Catalog. By using a crawler in AWS Glue, you can inspect a data source and infer its schema.

The team can now use crawlers to inspect data sources quickly and reduce the manual steps to create database schemas from data sources in Amazon S3. You share the results of this POC with Mary, and she is happy with the new functionality. Next, she wants to be able to do more analysis on the data in the AWS Glue Data Catalog.

Task 2: Querying a table by using Athena

Now that you created the Data Catalog, you can use the metadata to query the data even further by using Athena.

In this task, you will complete the following steps:

- Configure an S3 bucket to store Athena query results.
- Preview a database table in Athena.
- Create a table for data after 1950.
- Run a query on selected data.

7. Configure an S3 bucket to store Athena query results.
    - In the navigation pane, under **Databases**, choose **Tables**.
    - Choose the link for the **by_year** table.
    - Choose **Actions** > **View data**.
    - When the pop-up appears to warn you that you will be taken to the Athena console, choose **Proceed**.

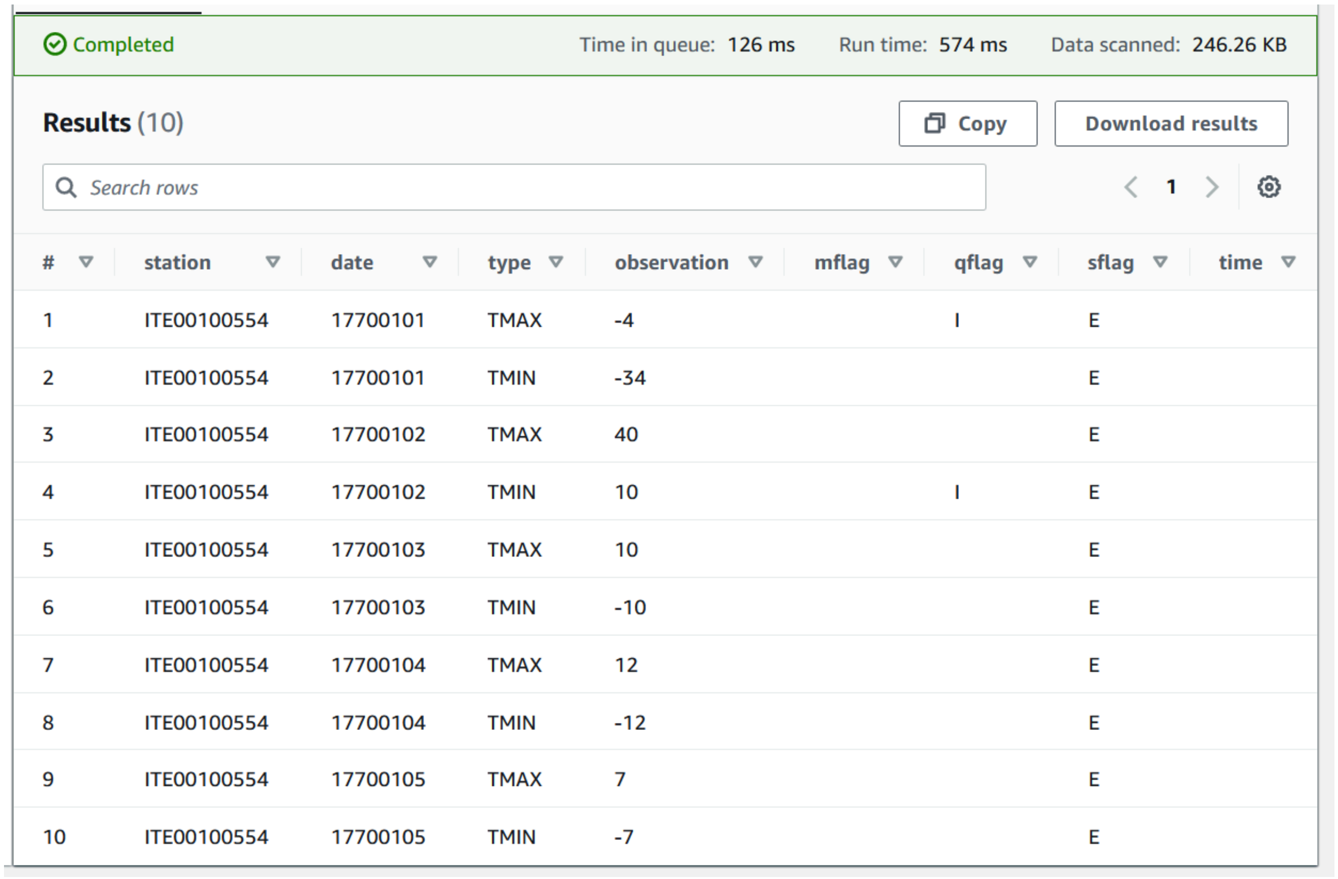
      The Athena console opens. Notice the error message that indicates that an output location was not provided. Before you run a query in Athena, you need to specify an S3 bucket to hold query results.
    - Choose the **Settings** tab.
    - Choose **Manage**.
    - To the right of **Location of query result**, choose **Browse S3**.
    - Choose the bucket name that is similar to the following: **data-science-bucket-XXXXXX**

      Important: Don't choose the bucket name that contains **glue-1950-bucket**.
    - Select **Choose**.
    - Keep the default settings for the other options, and choose **Save**.

8. Preview a table in Athena.
    - Choose the **Editor** tab.
    - In the **Data** panel on the left, notice that the **Data source** is **AwsDataCatalog**.
    - For **Database**, choose **weatherdata**.
    - In the **Tables** section, choose the ellipsis (three dot) icon for the **by_year** table, and then choose **Preview Table**.

      Tip: To view the column names and their data types in this table, choose the icon to the left of the table name.

      The first 10 records from the *weatherdata* table display, similar to the following screenshot:



Completed Time in queue: 126 ms Run time: 574 ms Data scanned: 246.26 KB

Results (10)

#	station	date	type	observation	mflag	qflag	sflag	time
1	ITE00100554	17700101	TMAX	-4		I	E	
2	ITE00100554	17700101	TMIN	-34			E	
3	ITE00100554	17700102	TMAX	40			E	
4	ITE00100554	17700102	TMIN	10		I	E	
5	ITE00100554	17700103	TMAX	10			E	
6	ITE00100554	17700103	TMIN	-10			E	
7	ITE00100554	17700104	TMAX	12			E	
8	ITE00100554	17700104	TMIN	-12			E	
9	ITE00100554	17700105	TMAX	7			E	
10	ITE00100554	17700105	TMIN	-7			E	

Notice the run time and amount of data that was scanned for the query. As you develop more complex applications, it is important to minimize resource consumption to optimize costs. You will see examples of how to optimize cost for Athena queries later in this task.

In the next step, you will create an AWS Glue database table that only includes data since 1950. To optimize your use of Athena, you will store data in the Apache Parquet format. Apache Parquet is an open-source columnar data format that is optimized for performance and storage.

9. Create a table for data after 1950.

   First, you need to retrieve the name of the bucket that was created for you to store this data.

   - In the search box next to **Services**, search for and choose **S3**.
   - In the **Buckets** list, copy the bucket name that contains **glue-1950-bucket** to a text editor of your choice.
   - Return to the Athena query editor.
   - Copy and paste the following query into a query tab in the editor. Replace *<glue-1950-bucket>* with the name of the bucket that you recorded:

```
CREATE table weatherdata.late20th
WITH (
  format='PARQUET', external_location='s3://<glue-1950-bucket>/lab3'
) AS SELECT date, type, observation  FROM by_year
WHERE date/10000 between 1950 and 2015;
```

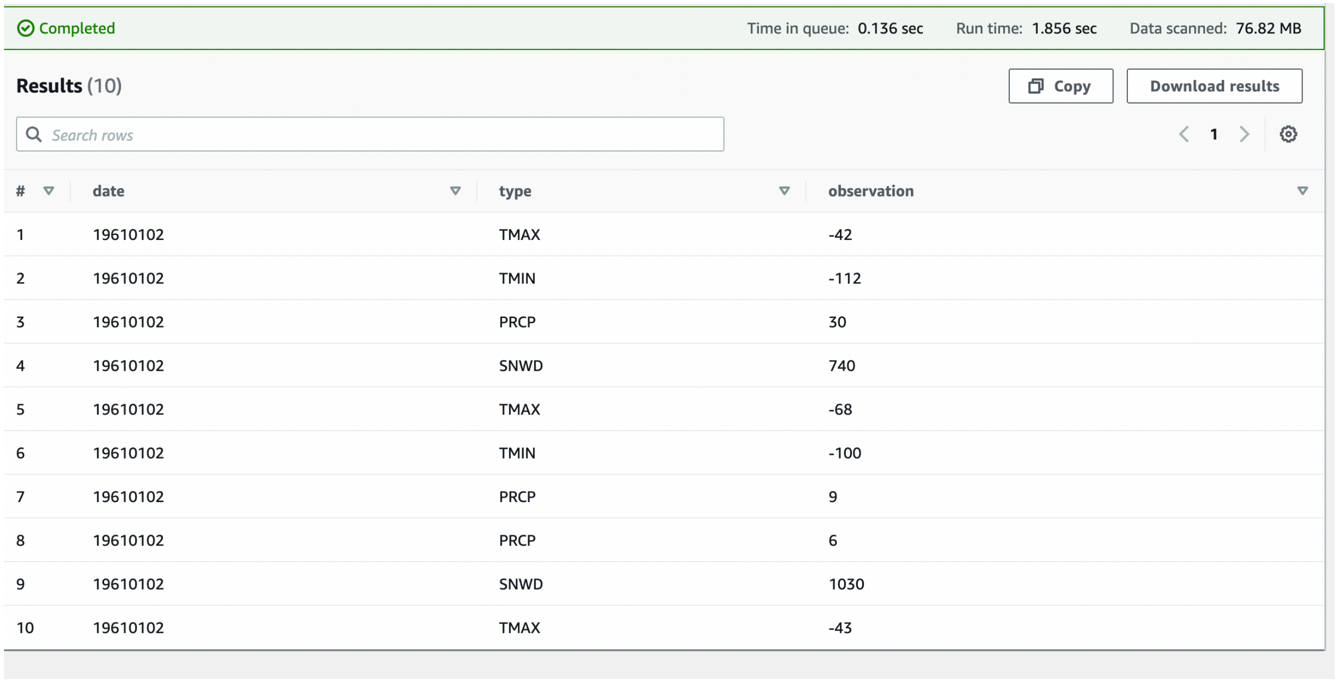
   - Choose **Run**.

     After the query runs, the run time and data scanned values are similar to the following:

```
Time in queue: 128 ms
Run time: 1 min 8.324 sec
Data scanned: 98.44 GB
```

- To preview the results, in the **Tables** section, to the right of the *late20th* table, choose the ellipsis icon, and then choose **Preview Table**.

  The results are similar to the following screenshot.



Completed Time in queue: 0.136 sec Run time: 1.856 sec Data scanned: 76.82 MB

Results (10)

#	date	type	observation
1	19610102	TMAX	-42
2	19610102	TMIN	-112
3	19610102	PRCP	30
4	19610102	SNWD	740
5	19610102	TMAX	-68
6	19610102	TMIN	-100
7	19610102	PRCP	9
8	19610102	PRCP	6
9	19610102	SNWD	1030
10	19610102	TMAX	-43

  Now that you have isolated the data that you are interested in, you can write queries for further analysis.

10. Run a query on the new table.

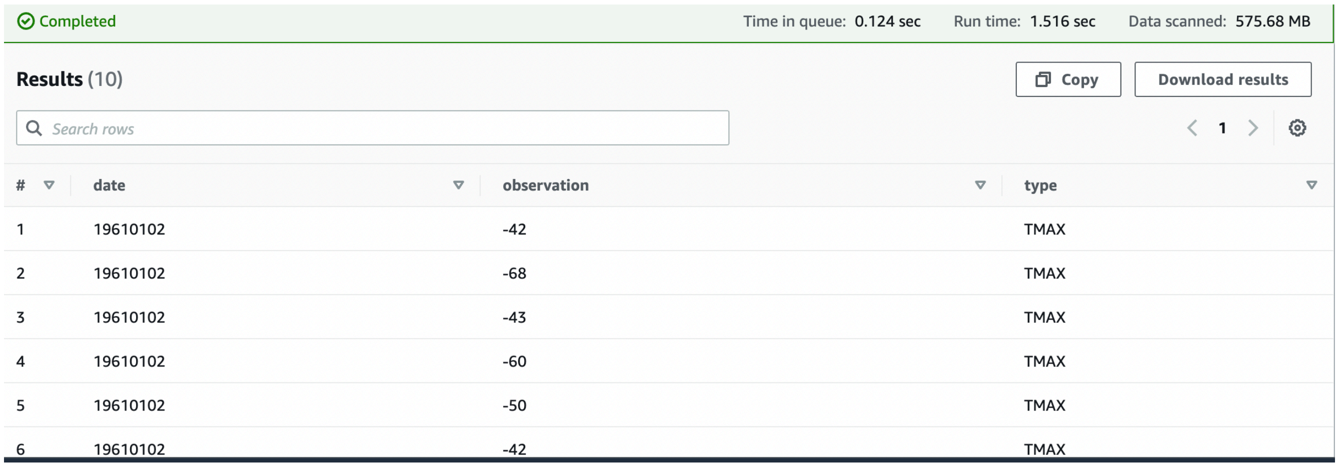
    First, create a view that only includes the maximum temperature reading, or TMAX, value.

    - Run the following query in a new query tab:

```
CREATE VIEW TMAX AS
SELECT date, observation, type
FROM late20th
WHERE type = 'TMAX'
```

    - To preview the results, in the **Views** section, to the right of the *tmax* view, choose the ellipsis icon, and then choose **Preview View**.

      The results are similar to the following screenshot:



Completed Time in queue: 0.124 sec Run time: 1.516 sec Data scanned: 575.68 MB

Results (10)

#	date	observation	type
1	19610102	-42	TMAX
2	19610102	-68	TMAX
3	19610102	-43	TMAX
4	19610102	-60	TMAX
5	19610102	-50	TMAX
6	19610102	-42	TMAX

11. Run the following query in a new query tab.

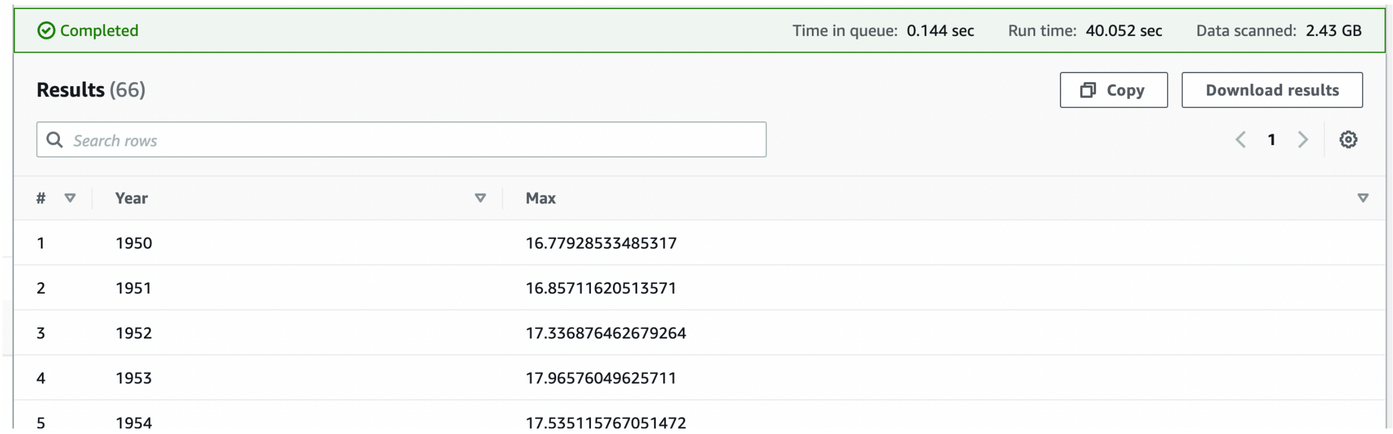
```
SELECT date/10000 as Year, avg(observation)/10 as Max
FROM tmax
GROUP BY date/10000 ORDER BY date/10000;
```

The purpose of this query is to calculate the average maximum temperature for each year in the dataset.

After the query runs, the run time and data scanned values are similar to the following:

```
Time in queue: 0.211 sec
Run time: 25.109 sec
Data scanned: 2.45 GB
```

The results display the average maximum temperature for each year from 1950 to 2015. The following screenshot displays an example:



Completed Time in queue: 0.144 sec Run time: 40.052 sec Data scanned: 2.43 GB

Results (66)

#	Year	Max
1	1950	16.77928533485317
2	1951	16.85711620513571
3	1952	17.336876462679264
4	1953	17.96576049625711
5	1954	17.535115767051472

Remember that when you create queries with Athena, the results of the query must be stored back in Amazon S3. In a previous step, you specified the location in Amazon S3 where your queries are stored.

When using AWS services, you generally pay for what you use. However, because you reduced your query to only three columns of temperature data from 1950 through 2015, you reduced your costs for storage. Also, because you arranged the query data in columnar format with Apache Parquet, the time that it took to perform the queries in this task was reduced, resulting in less cost as you used fewer computational resources in Athena.

You show Mary that she can speed up her process by using AWS Glue in combination with Athena and still use views. She is delighted.

Task 2 summary

In this task, you learned how to use Athena to query tables in a database that an AWS Glue crawler created. You built a table for all data after 1950 from the original dataset. You used the Apache Parquet format to optimize your Athena queries, which reduced the time that it took to complete each query, resulting in less cost. After isolating this data, you created a view that calculated the average maximum temperature for each year.

AWS Glue integrates with original datasets stored in Amazon S3. AWS Glue can create a crawler to ingest the original dataset into a database and infer the appropriate schema. Then, you can quickly shift to Athena to develop queries and better understand your data. This integration reduces the time that it takes to derive insights from your data and apply these insights to make better decisions.

Task 3: Creating a CloudFormation template for an AWS Glue crawler

In task 1, you used the console to create an AWS Glue crawler to inspect the data source and infer a schema. However, the team works with many datasets in different AWS accounts, including development, test, and production. Therefore, it would be helpful to reuse crawlers across these environments, especially when new data is added to the datasets. It would also be helpful to use the AWS CLI to run the crawler.

If your crawler runs more than once, perhaps on a schedule, it looks for new or changed files or tables in your data store. The output of the crawler includes new tables and partitions that were found since a previous run.

In this task, you will learn how to create and deploy a crawler by using CloudFormation.

12. Find the Amazon Resource Number (ARN) for the *gluelab* IAM role. You need this ARN to deploy the CloudFormation template.
    - In the search box next to **Services**, search for and choose **IAM** to open the IAM console.
    - In the navigation pane, choose **Roles**.
    - Choose the link for the **gluelab** role.

      Tip: You can search for the role if needed.

      The ARN is displayed on the page in the **Summary** section.
    - Copy the ARN to a text editor to use in the next step.
13. Navigate to the AWS Cloud9 integrated development environment (IDE).

- In the search box next to **Services**, search for and choose **Cloud9** to open the AWS Cloud9 console.

  AWS Cloud9 environments are listed.
- For the environment named **Cloud9 Instance**, choose **Open IDE**.

  A new browser tab opens and displays the AWS Cloud9 IDE.

14. Create a new CloudFormation template.
    - In the AWS Cloud9 IDE, choose **File** > **New File**.
    - Save the empty file as `gluecrawler.cf.yml` but keep it open.
    - Copy and paste the following code into the file:

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
        # The name of the crawler to be created
 CFNCrawlerName:
   Type: String
   Default: cfn-crawler-weather
 CFNDatabaseName:
   Type: String
   Default: cfn-database-weather
 CFNTablePrefixName:
   Type: String
   Default: cfn_sample_1-weather
# Resources section defines metadata for the Data Catalog
Resources:
# Create a database to contain tables created by the crawler
 CFNDatabaseWeather:
   Type: AWS::Glue::Database
   Properties:
     CatalogId: !Ref AWS::AccountId
     DatabaseInput:
       Name: !Ref CFNDatabaseName
       Description: "AWS Glue container to hold metadata tables for the
weather crawler"
#Create a crawler to crawl the weather data on a public S3 bucket
 CFNCrawlerWeather:
   Type: AWS::Glue::Crawler
   Properties:
     Name: !Ref CFNCrawlerName
     Role: <GLUELAB-ROLE-ARN>
     #Classifiers: none, use the default classifier
     Description: AWS Glue crawler to crawl weather data
     #Schedule: none, use default run-on-demand
     DatabaseName: !Ref CFNDatabaseName
     Targets:
       S3Targets:
         # Public S3 bucket with the weather data
         - Path: "s3://noaa-ghcn-pds/csv/by_year/"
     TablePrefix: !Ref CFNTablePrefixName
     SchemaChangePolicy:
       UpdateBehavior: "UPDATE_IN_DATABASE"
       DeleteBehavior: "LOG"
     Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":
{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}"
```

Note: This codeblock uses the **Sample AWS CloudFormation Template for an AWS Glue Crawler for Amazon S3** from [AWS CloudFormation for AWS Glue]() in the *AWS Glue Developer Guide*.

- In the file, replace <*GLUELAB-ROLE-ARN*> with the ARN for the *gluelab* IAM role. Look for the line that starts with **Role**, which is around line 27.
- Save the changes to the template file.

  Examine the code to see what is being created. This CloudFormation template does the following:

  - Sets a custom resource name for the AWS Glue crawler.
  - Sets a custom resource name for the AWS Glue database.
  - Sets a custom resource name for the first table in the AWS Glue database.
  - Creates a crawler to crawl the weather data on a public S3 bucket.
  - Sets the IAM role (*gluelab*) that the crawler will use to create the associated AWS Glue database. This is the same role that you used to create the crawler manually. Note that this IAM role was created for you in the lab environment. The role has all of the permissions that are necessary to create and run the crawler and create the database.

15. To validate the CloudFormation template, run the following command in the AWS Cloud9 terminal:

```
aws cloudformation validate-template --template-body
file://gluecrawler.cf.yml
```

Note: If you receive an error that says *YAML not well-formed*, check the value for the name of the *gluelab* role. Also check the tabs and spacing for each line. YAML documents require exact spacing, and the parser will encounter errors if the spacing doesn't match.

If the template is validated, the following output displays:

```
{
    "Parameters": [
        {
            "ParameterKey": "CFNCrawlerName",
            "DefaultValue": "cfn-crawler-weather",
            "NoEcho": false
        },
        {
            "ParameterKey": "CFNTablePrefixName",
            "DefaultValue": "cfn_sample_1-weather",
            "NoEcho": false
        },
        {
            "ParameterKey": "CFNDatabaseName",
            "DefaultValue": "cfn-database-weather",
            "NoEcho": false
        }
    ]
}
```

Important: Don't go to the next step until the template is validated.

Now you will use the template to create a CloudFormation stack. A *stack* implements and manages the group of resources that are outlined in a template. With a stack, you can manage the state and dependencies of those resources together. Think of a CloudFormation template as a blueprint. Then, the stack is the actual instance of the template that is registered in AWS and actually creates the resources.

16. To create the CloudFormation stack, run the following command:

```
aws cloudformation create-stack --stack-name gluecrawler --template-body
file://gluecrawler.cf.yml --capabilities CAPABILITY_NAMED_IAM
```

Note: The command includes the --capabilities parameter with the CAPABILITY_NAMED_IAM capability. This is because you are creating the following resources with custom names, which affect permissions:

- An AWS Glue crawler named *cfn-crawler-weather*
- An AWS Glue database named *cfn-database-weather*
- A table named *cfn_sample_1-weather* within the AWS Glue database

If the stack is validated, the CloudFormation ARN displays in the output, similar to the following:

```
{
"StackId": "arn:aws:cloudformation:us-east-
1:338778555682:stack/gluecrawler/2d8cec90-5c42-11ec-8fbf-12034b0079a5"
}
```

The CloudFormation create-stack command creates the stack and deploys it. If validation passes and nothing causes the stack creation to roll back, proceed to the next step.

Tip: To check the progress of stack creation, navigate to the CloudFormation console. In the navigation pane, choose **Stacks**.

17. To verify that the AWS Glue database was created in the stack, run the following command:

```
aws glue get-databases
```

The output is similar to the following:

```
{
    "DatabaseList": [
        {
            "Name": "cfn-database-weather",
            "Description": "AWS Glue container to hold metadata tables for
the weather crawler",
            "Parameters": {},
            "CreateTime": 1649267047.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier":
"IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "034140262343"
        },
        {
            "Name": "weatherdata",
            "CreateTime": 1649263434.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier":
"IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "034140262343"
        }
    ]
}
```

18. Verify that the crawler was created in the stack.
    - To verify that the crawler was created, run the following command:

```
aws glue list-crawlers
```

The output is similar to the following:

```
{
    "CrawlerNames": [
        "Weather",
        "cfn-crawler-weather"
    ]
}
```

- To retrieve the details of the crawler, run the following command.

```
aws glue get-crawler --name cfn-crawler-weather
```

The output is similar to the following:

```
{
    "Crawler": {
       "Name": "cfn-crawler-weather",
        "Role": "WeatherCrawler-001-CFNRoleWeather-17WB9OM5H5MFL",
        "Targets": {
            "S3Targets": [
                {
                    "Path": "s3://noaa-ghcn-pds/csv/by_year/",
                    "Exclusions": []
                }
            ],
            "JdbcTargets": [],
            "MongoDBTargets": [],
            "DynamoDBTargets": [],
            "CatalogTargets": [],
            "DeltaTargets": []
        },
        "DatabaseName": "cfn-database-weather",
        "Description": "AWS Glue crawler to crawl weather data",
        "Classifiers": [],
        "RecrawlPolicy": {
            "RecrawlBehavior": "CRAWL_EVERYTHING"
        },
        "SchemaChangePolicy": {
            "UpdateBehavior": "UPDATE_IN_DATABASE",
            "DeleteBehavior": "LOG"
        },
        "LineageConfiguration": {
            "CrawlerLineageSettings": "DISABLE"
        },
        "State": "READY",
        "TablePrefix": "cfn_sample_1-weather",
        "CrawlElapsedTime": 0,
        "CreationTime": 1649083535.0,
        "LastUpdated": 1649083535.0,
        "Version": 1,
        "Configuration": "{\"Version\":1.0,\"CrawlerOutput\":
{\"Partitions\":{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":
{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}",
        "LakeFormationConfiguration": {
            "UseLakeFormationCredentials": false,
            "AccountId": ""
        }
    }
}
```

Review the response. Notice that the state of the crawler is *READY*. This means that the crawler is deployed, but it hasn't run yet. You will use Mary's IAM user to run the crawler later in the lab.

Task 3 summary

In this task, you learned how to integrate an AWS Glue crawler into a CloudFormation template. You also learned how to use the AWS CLI within the AWS Cloud9 terminal to validate and deploy the template to create the crawler. With the template, you can reuse the crawler in other AWS accounts. Then, you learned how to confirm that the resources were built with the crawler (the AWS Glue database and its associated tables).

Many companies use multiple accounts with AWS to maintain separate development, testing, and production environments. Isolating these environments helps to ensure that teams follow best practices. Building crawlers in a development account and then testing them in a controlled account with production data can help to ensure that the crawler is designed as intended and extracts, transforms, and loads the data that is intended for the specific business task. After validating the crawler, you can use DevOps best practices with CloudFormation to quickly move it to production so that the appropriate business stakeholders can reuse the crawler without having to build from the beginning.

Task 4: Reviewing the IAM policy for Athena and AWS Glue access

Now that you have created the crawler by using CloudFormation, review the IAM policy for the crawler to ensure that others can use it in production.

Note: The IAM policy for the crawler was created for you; you don't have the ability to create IAM policies in the lab environment.

19. Review the *Policy-For-Data-Scientists* policy in IAM.
    - In the search box to the right of **Services**, search for and choose **IAM** to open the IAM console.
    - In the navigation pane, choose **Users**.

      Note that *mary* is one of the IAM users that is listed. This user is part of the *DataScienceGroup* IAM group.
    - Choose the link for the **DataScienceGroup** IAM group.
    - On the *DataScienceGroup* details page, choose the **Permissions** tab.
    - In the list of policies that are attached to the group, choose the link for the **Policy-For-Data-Scientists** policy.

      The *Policy-For-Data-Scientists* details page opens. Review the permissions that are associated with this policy. Notice that the permissions provide limited access for only the Athena, AWS Glue, and Amazon S3 services.

      Tip: To look more closely at the details of the IAM policy, choose **{} JSON**. In the JSON policy file, you can see the allowed and denied actions, including which resources the users can perform actions on.

Task 4 summary

In this task, you reviewed the IAM policy for the *DataScienceGroup*. The policy contains permissions for limited access to Amazon S3, AWS Glue, and Athena. The policy could be used as an example policy for users who intend to reuse crawlers built by the operations team. As with all services in AWS, IAM users must have the appropriate permissions applied to be able to perform actions.

Task 5: Confirming that Mary can access and use the AWS Glue crawler

Now that you have reviewed the IAM policy, you will use it to test another user's access to the AWS Glue crawler. You will also test the user's ability to use the crawler to extract, transform, and load data from a dataset stored in Amazon S3 into an AWS Glue database.

20. Retrieve the credentials for the *mary* IAM user, and store these as bash variables.
    - In the search box next to **Services**, search for and choose **CloudFormation**.
    - In the navigation pane, choose **Stacks**.
    - Choose the link for the stack that created the lab environment. The stack name includes a random string of letters and numbers, and the stack should have the oldest creation time.
    - On the stack details page, choose the **Outputs** tab.

      Note: When you create a CloudFormation template, you can choose to output information about the resources that the template will create. The CloudFormation template that created the resources in your lab environment output the access key and secret access key for the *mary* user.
    - Copy the value of **MarysAccessKey** to your clipboard.
    - Return to the AWS Cloud9 terminal.

- To create a variable for the access key, run the following command. Replace *<ACCESS-KEY>* with the value from your clipboard.

```
AK=<ACCESS-KEY>
```

- Return to the CloudFormation console, and copy the value of **MarysSecretAccessKey** to your clipboard.
- Return to the AWS Cloud9 terminal.
- To create a variable for the secret access key, run the following command. Replace *<SECRET-ACCESS-KEY>* with the value from your clipboard.

```
SAK=<SECRET-ACCESS-KEY>
```

To test whether the *mary* user can perform a specific command, you can pass the user's credentials as bash variables (*AK* and *SAK*) with the command. The API will then try to perform that command as the specified user.

21. Test Mary's access to the AWS Glue crawler.
    - To test whether the *mary* user can perform the list-crawlers command, run the following command:

```
AWS_ACCESS_KEY_ID=$AK AWS_SECRET_ACCESS_KEY=$SAK aws glue list-crawlers
```

The output is similar to the following and looks like the output that was displayed after you ran the command earlier:

```
{
    "CrawlerNames": [
        "weather",
        "cfn-crawler-weather"
    ]
}
```

- To test whether the *mary* user can perform the get-crawler command, run the following command:

```
AWS_ACCESS_KEY_ID=$AK AWS_SECRET_ACCESS_KEY=$SAK aws glue get-crawler --
name cfn-crawler-weather
```

The output is similar to the following and looks like the output that was displayed after you ran the command earlier. Note that the state of the crawler is *READY*, but no status information is displayed. This is because the crawler hasn't run yet.

```
{
    "Crawler": {
        "Name": "cfn-crawler-weather",
        "Role": "gluelab",
        "Targets": {
            "S3Targets": [
                {
                    "Path": "s3://noaa-ghcn-pds/csv/by_year/",
                    "Exclusions": []
                }
            ],
            "JdbcTargets": [],
            "MongoDBTargets": [],
            "DynamoDBTargets": [],
            "CatalogTargets": [],
            "DeltaTargets": []
        },
        "DatabaseName": "cfn-database-weather",
        "Description": "AWS Glue crawler to crawl weather data",
        "Classifiers": [],
        "RecrawlPolicy": {
            "RecrawlBehavior": "CRAWL_EVERYTHING"
        },
        "SchemaChangePolicy": {
            "UpdateBehavior": "UPDATE_IN_DATABASE",
            "DeleteBehavior": "LOG"
        },
        "LineageConfiguration": {
            "CrawlerLineageSettings": "DISABLE"
        },
        "State": "READY",
        "TablePrefix": "cfn_sample_1-weather",
        "CrawlElapsedTime": 0,
        "CreationTime": 1649267047.0,
        "LastUpdated": 1649267047.0,
        "Version": 1,
        "Configuration": "{\"Version\":1.0,\"CrawlerOutput\":
{\"Partitions\":{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":
{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}",
        "LakeFormationConfiguration": {
            "UseLakeFormationCredentials": false,
            "AccountId": ""
        }
    }
}
```

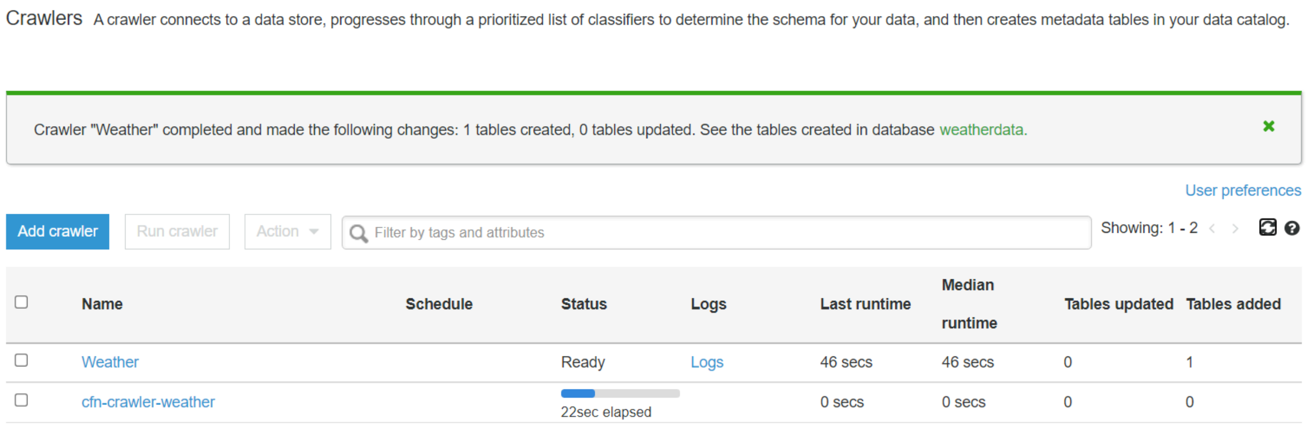
22. Test that the *mary* user can run the crawler.
    - Run the following command.

```
AWS_ACCESS_KEY_ID=$AK AWS_SECRET_ACCESS_KEY=$SAK aws glue start-crawler --
name cfn-crawler-weather
```

If the crawler runs successfully, the terminal doesn't display any output.

- To observe the crawler running and adding data to the table, navigate to the AWS Glue console.
- In the navigation pane, choose **Crawlers**.

  Here you can see status information for the crawler, as shown in the following screenshot.



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.

Crawler "Weather" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database weatherdata.

	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
	Weather		Ready	Logs	46 secs	46 secs	0	1
	cfn-crawler-weather		22sec elapsed		0 secs	0 secs	0	0

  When the status changes to *Ready*, the crawler is finished running. It might take a few minutes.

  Return to the AWS Cloud9 terminal.

- To confirm that the crawler is finished running, run the following command.

```
AWS_ACCESS_KEY_ID=$AK AWS_SECRET_ACCESS_KEY=$SAK aws glue get-crawler --
name cfn-crawler-weather
```

  The output is similar to the following:

```
{
    "Crawler": {
        "Name": "cfn-crawler-weather",
        "Role": "gluelab",
        "Targets": {
            "S3Targets": [
                {
                    "Path": "s3://noaa-ghcn-pds/csv/by_year/",
                    "Exclusions": []
                }
            ],
            "JdbcTargets": [],
            "MongoDBTargets": [],
            "DynamoDBTargets": [],
            "CatalogTargets": [],
            "DeltaTargets": []
        },
        "DatabaseName": "cfn-database-weather",
        "Description": "AWS Glue crawler to crawl weather data",
        "Classifiers": [],
        "RecrawlPolicy": {
            "RecrawlBehavior": "CRAWL_EVERYTHING"
        },
        "SchemaChangePolicy": {
            "UpdateBehavior": "UPDATE_IN_DATABASE",
            "DeleteBehavior": "LOG"
        },
        "LineageConfiguration": {
            "CrawlerLineageSettings": "DISABLE"
        },
        "State": "READY",
        "TablePrefix": "cfn_sample_1-weather",
        "CrawlElapsedTime": 0,
        "CreationTime": 1649267047.0,
        "LastUpdated": 1649267047.0,
        "LastCrawl": {
            "Status": "SUCCEEDED",
            "LogGroup": "/aws-glue/crawlers",
            "LogStream": "cfn-crawler-weather",
            "MessagePrefix": "5ef3cff5-ce6c-45d5-8359-e223a4227570",
            "StartTime": 1649267649.0
        },
        "Version": 1,
        "Configuration": "{\"Version\":1.0,\"CrawlerOutput\":
{\"Partitions\":{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":
{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}",
        "LakeFormationConfiguration": {
            "UseLakeFormationCredentials": false,
            "AccountId": ""
        }
    }
}
```

Notice that the **LastCrawl** section is included, and the status in that section is *SUCCEEDED*. This means that Mary was able to run the crawler successfully.

Task 5 summary

This result confirms that Mary has access to the AWS Glue crawler that you created and deployed with CloudFormation. This is because the permissions in the IAM policy allow her to list, get, and retrieve the metadata for the crawler. Other permissions associated with the policy include the following:

- **For AWS Glue:** List, read, and tag resources. Run a crawler deployed with CloudFormation, but not create or remove resources or manage.
- **For Athena:** List, read, and tag resources, but not create or remove *specific* resources. (For example, this policy does not provide permissions to create or remove a named query or Data Catalog, which your user has permissions to do.)
- **For Amazon S3:** Access buckets, list bucket contents, and read objects, but not create a bucket and limit access to specific buckets, like the DataScienceBucket we created for you.

Congratulations! You have learned how to create an AWS Glue crawler manually and by using CloudFormation so that you can deploy it to users with a secure IAM policy. Because the crawler is in a CloudFormation template, you can reuse the template to create and deploy the crawler in any AWS account and change the parameters as desired.

Update from the team

The team is happy with what you have learned and demonstrated by using Athena, AWS Glue, and CloudFormation. Now that you have shared what you have learned, the team will be able to simplify their workloads and use AWS in a way that follows best practices.

Submitting your work

23. To record your progress, choose **Submit** at the top of these instructions.
24. When prompted, choose **Yes**.

    After a couple of minutes, the grades panel appears and shows you how many points you earned for each task. If the results don't display after a couple of minutes, choose **Grades** at the top of these instructions.

    Important: Some of the checks made by the submission process in this lab will only give you credit if it has been at least 5 minutes since you completed the action. If you do not receive credit the first time you submit, you may need to wait a couple minutes and the submit again to receive credit for these items.

    Tip: You can submit your work multiple times. After you change your work, choose **Submit** again. Your last submission is recorded for this lab.
25. To find detailed feedback about your work, choose **Submission Report**.

Lab complete

Congratulations! You have completed the lab.

26. At the top of this page, choose **End Lab**, and then choose **Yes** to confirm that you want to end the lab.

    A message panel indicates that the lab is terminating.

27. To close the panel, choose **Close** in the upper-right corner.

Additional resources

For more information about the services and concepts covered in this lab, see the following resources:

- Getting Started with AWS Glue
- Scheduling an AWS Glue Crawler
- Apache Parquet
- How Crawlers Work
- CreateStack Request in CloudFormation