
AWS Glue Studio

User Guide



AWS Glue Studio: User Guide

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Glue Studio?	1
Features of AWS Glue Studio	2
Visual job editor	2
Job script code editor	2
Job performance dashboard	3
Support for dataset partitioning	3
When should I use AWS Glue Studio?	3
Accessing AWS Glue Studio	3
Pricing for AWS Glue Studio	4
Setting up	5
Sign up for AWS	5
Create an IAM administrator user	5
Signing in as an IAM user	6
IAM permissions needed for the AWS Glue Studio user	6
AWS Glue service permissions	6
Data preview permissions	7
Amazon CloudWatch permissions	7
Overview of Job-related permissions	8
Data source and data target permissions	8
Permissions required for deleting jobs	8
AWS Key Management Service permissions	8
Permissions required for using connectors	9
Set up IAM permissions for AWS Glue Studio	9
Configuring a VPC for your ETL job	9
Populate the AWS Glue Data Catalog	10
Tutorial: Getting started	11
Prerequisites	11
Step 1: Start the job creation process	11
Step 2: Edit the data source node in the job diagram	12
Step 3: Edit the transform node of the job	13
Step 4: Edit the data target node of the job	13
Step 5: View the job script	14
Step 6: Specify the job details and save the job	14
Step 7: Run the job	15
Next steps	15
Creating jobs	16
Start the job creation process	16
Create jobs that use a connector	17
Next steps for creating a job in AWS Glue Studio	17
Editing jobs	18
Accessing the job diagram editor	18
Job editor features	18
Using schema previews in the visual job editor	19
Using data previews in the visual job editor	19
Restrictions when using data previews	20
Editing the data source node	20
Using Data Catalog tables for the data source	21
Using a connector for the data source	22
Using files in Amazon S3 for the data source	22
Using a streaming data source	23
Editing the data transform node	25
Overview of mappings and transforms	26
Using ApplyMapping to remap data property keys	26
Using SelectFields to remove most data property keys	27

Using DropFields to keep most data property keys	27
Renaming a field in the dataset	28
Using Spigot to sample your dataset	29
Joining datasets	29
Using SplitFields to split a dataset into two	31
Overview of <i>SelectFromCollection</i> transform	31
Using SelectFromCollection to choose which dataset to keep	32
Filtering keys within a dataset	32
Find and fill missing values in a dataset	33
Using a SQL query to transform data	34
Creating a custom transformation	35
Configuring data target nodes	38
Overview of data target options	38
Editing the data target node	39
Editing or uploading a job script	41
Creating and editing Scala scripts in AWS Glue Studio	42
Creating and editing Python shell jobs in AWS Glue Studio	43
Adding nodes to the job diagram	44
Changing the parent nodes for a node in the job diagram	44
Deleting nodes from the job diagram	45
Using connectors and connections	46
Overview of using connectors and connections	46
Adding connectors to AWS Glue Studio	47
Subscribing to AWS Marketplace connectors	47
Creating custom connectors	48
Creating connections for connectors	50
Authoring jobs with custom connectors	50
Create jobs that use a connector for the data source	50
Configure source properties for nodes that use connectors	51
Configure target properties for nodes that use connectors	54
Managing connectors and connections	55
Viewing connector and connection details	55
Editing connectors and connections	56
Deleting connectors and connections	56
Cancel a subscription for a connector	56
Developing custom connectors	57
Developing Spark connectors	57
Developing Athena connectors	57
Developing JDBC connectors	58
Examples of using custom connectors with AWS Glue Studio	58
Developing AWS Glue connectors for AWS Marketplace	58
Restrictions for using connectors and connections in AWS Glue Studio	59
Tutorial: Using the open-source Elasticsearch Spark Connector	60
Prerequisites	60
Step 1: (Optional) Create an AWS secret for your OpenSearch cluster information	60
Next step	61
Step 2: Subscribe to the connector	61
Next step	62
Step 3: Activate the connector in AWS Glue Studio and create a connection	62
Next step	62
Step 4: Configure an IAM role for your ETL job	62
Next step	62
Step 5: Create a job that uses the OpenSearch connection	63
Next step	65
Step 6: Run the job	65
Managing jobs	66
Start a job run	66

Schedule job runs	66
Manage job schedules	67
Stop job runs	68
View your jobs	68
Customize the job display	68
View information for recent job runs	68
View the job script	69
Modify the job properties	69
Store Spark shuffle files on Amazon S3	70
Save the job	71
Troubleshooting errors when saving a job	71
Clone a job	73
Delete jobs	73
Monitoring jobs	74
Accessing the job monitoring dashboard	74
Overview of the job monitoring dashboard	74
Job runs view	74
Viewing the job run logs	76
Viewing the details of a job run	76
Viewing Amazon CloudWatch metrics for a job run	77
Tutorial: Adding an AWS Glue crawler	79
Prerequisites	79
Step 1: Add a crawler	79
Step 2: Run the crawler	80
Step 3: View AWS Glue Data Catalog objects	80
Document history	82
AWS glossary	86

What is AWS Glue Studio?

AWS Glue Studio is a new graphical interface that makes it easy to create, run, and monitor extract, transform, and load (ETL) jobs in AWS Glue. You can visually compose data transformation workflows and seamlessly run them on AWS Glue's Apache Spark-based serverless ETL engine. You can inspect the schema and data results in each step of the job.

The screenshot displays the AWS Glue Studio interface for a job titled "Combine legislator data". The workflow is visualized in the center, showing a sequence of steps: two data sources (Memberships and Organizations) are joined, then transformed (ApplyMapping), and finally written out as a JSON file. The right-hand panel shows the "Data preview" for the current step, displaying a table of legislator data.

family_name	name	gender	birth_date	death_date
Collins	Mac Collins	male	1944-10-15	null
Huizenga	Bill Huizenga	male	1969-01-31	null
Clawson	Curt Clawson	male	1959-09-28	null
Solomon	Gerald Solomon	male	1930-08-14	2001-10-26
Rigell	E. Scott Rigell	male	1960-05-28	null
Crapo	Mike Crapo	male	1951-05-20	null
Hutto	Earl Hutto	male	1926-05-12	null
Ertel	Allen Ertel	male	1937-11-07	2015-11-19
Minish	Joseph Minish	male	1916-09-01	2007-11-24
Andrews	Robert E. Andrews	male	1957-08-04	null
Walden	Greg Walden	male	1957-01-10	null
Kazen	Abraham Kazen, Jr.	male	1919-01-17	1987-11-29
Turner	Michael R. Turner	male	1960-01-11	null
Kolbe	Jim Kolbe	male	1942-06-28	null
Lowenthal	Alan S. Lowenthal	male	1941-03-08	null
Capuano	Michael E. Capuano	male	1952-01-09	null
Schrader	Kurt Schrader	male	1951-10-19	null
Nadler	Jerrold Nadler	male	1947-06-13	null
Graves	Tom Graves	male	1970-02-03	null
McMillan	John McMillan	male	1932-05-09	null

AWS Glue Studio is designed not only for tabular data, but also for semi-structured data, which is difficult to render in spreadsheet-like data preparation interfaces. Examples of semi-structured data include application logs, mobile events, Internet of Things (IoT) event streams, and social feeds.

When creating a job in AWS Glue Studio, you can choose from a variety of data sources that are stored in AWS services. You can quickly prepare that data for analysis in data warehouses and data lakes. AWS Glue Studio also offers tools to monitor ETL workflows and validate that they are operating as intended. You can preview the dataset for each node. This helps you to debug your ETL jobs by displaying a sample of the data at each step of the job.

AWS Glue Studio provides a visual interface that makes it easy to:

- Pull data from an Amazon S3, Amazon Kinesis, or JDBC source.
- Configure a transformation that joins, samples, or transforms the data.
- Specify a target location for the transformed data.

- View the schema or a sample of the dataset at each point in the job.
- Run, monitor, and manage the jobs created in AWS Glue Studio.

Features of AWS Glue Studio

AWS Glue Studio helps you to create and manage jobs that gather, transform, and clean data. Advanced users can use AWS Glue Studio to troubleshoot and edit job scripts.

Visual job editor

You can perform the following actions when creating and editing jobs in AWS Glue Studio:

- Add additional nodes to the job to implement:
 - Multiple data sources.
 - Multiple data targets.
 - Data sources and targets that use connectors for external data stores that were not previously supported
- View a sample of the data at each node in the job diagram.
- Change the parent nodes for an existing node.
- Add transforms that:
 - Join data sources.
 - Select specific fields from the data.
 - Drop fields.
 - Rename fields.
 - Change the data type of fields.
 - Write select fields from the data into a JSON file in an Amazon S3 bucket (spigot).
 - Filter out data from a dataset.
 - Split a dataset into two datasets.
 - Find missing values in a dataset and provide the missing value in a separate column.
 - Use SQL to query and transform the data.
 - Use custom code.

Job script code editor

AWS Glue Studio also has a script editor for writing or customizing the extract-transform-and-load (ETL) code for your jobs. You can use the visual editor in AWS Glue Studio to quickly design your ETL job and then edit the generated script to write code for the unique components of your job.

When creating a new job, you can choose to write scripts for Spark jobs or Python shell jobs. You can code the job ETL script for Spark jobs using either Python or Scala. If you create a Python shell job, the job ETL script uses Python 3.6.

The script editor interface in AWS Glue Studio offers the following features:

- Insert, modify, and delete sources, targets, and transforms in your script.
- Add or modify arguments for data sources, targets, and transforms.
- Syntax and keyword highlighting

- Auto-completion suggestions for local words, Python keywords, and code snippets.

Job performance dashboard

AWS Glue Studio provides a comprehensive run dashboard for your ETL jobs. The dashboard displays information about job runs from a specific time frame. The information displayed on the dashboard includes:

- Jobs overview summary – A high-level overview showing total jobs, current runs, completed runs, and failed jobs.
- Status summaries – Provides high level job metrics based on job properties, such as worker type and job type.
- Job runs time line – A bar graph summary of successful, failed, and total runs for the currently selected time frame.
- Job run breakdown – A detailed list of job runs from the selected time frame.

Support for dataset partitioning

You can use AWS Glue Studio to efficiently process partitioned datasets. You can load, filter, transform, and save your partitioned data by using SQL expressions or user-defined functions—to avoid listing and reading unnecessary data from Amazon S3.

When should I use AWS Glue Studio?

Use AWS Glue Studio for a simple visual interface to create ETL workflows for data cleaning and transformation, and run them on AWS Glue.

AWS Glue Studio makes it easy for ETL developers to create repeatable processes to move and transform large-scale, semi-structured datasets, and load them into data lakes and data warehouses. It provides a boxes-and-arrows style visual interface for developing and managing AWS Glue ETL workflows that you can optionally customize with code. AWS Glue Studio combines the ease of use of traditional ETL tools, and the power and flexibility of AWS Glue's big data processing engine.

AWS Glue Studio provides multiple ways to customize your ETL scripts, including adding nodes that represent code snippets in the visual editor.

Use AWS Glue Studio for easier job management. AWS Glue Studio provides you with job and job run management interfaces that make it clear how jobs relate to each other, and give an overall picture of your job runs. The job management page makes it easy to do bulk operations on jobs (previously difficult to do in the AWS Glue console). All job runs are available in a single interface where you can search and filter. This gives you a constantly updated view of your ETL operations and the resources you use. You can use the real-time dashboard in AWS Glue Studio to monitor your job runs and validate that they are operating as intended.

Accessing AWS Glue Studio

To access AWS Glue Studio, sign in to AWS as a user that has the required permissions, as described in [Set up IAM permissions for AWS Glue Studio \(p. 9\)](#). Then you can sign in to the AWS Management Console and open the AWS Glue console at <https://console.aws.amazon.com/glue/>. Click the **AWS Glue Studio** link in the navigation pane.

Pricing for AWS Glue Studio

When using AWS Glue Studio, you are charged for data previews. After you specify an IAM role for the job, the visual editor starts an Apache Spark session for sampling your source data and executing transformations. This session runs for 30 minutes and then turns off automatically. AWS charges you for 2 DPU-hours at the development endpoint rate (DEVED-DPU-Hour), typically resulting in a charge of \$0.44 for each 30 minute session. The rate might vary for each region. At the end of the 30 minute session, you can choose **Retry** on the **Data preview** tab for any node or reload the visual editor page to start a new 30 minute session at the same rates.

You also pay for the underlying AWS services that your jobs use or interact with—for example, AWS Glue, your data sources, and your data targets. For pricing information, see [AWS Glue Pricing](#).

Setting up for AWS Glue Studio

Complete the tasks in this section when you're using AWS Glue Studio for the first time:

Topics

- [Sign up for AWS \(p. 5\)](#)
- [Create an IAM administrator user \(p. 5\)](#)
- [Signing in as an IAM user \(p. 6\)](#)
- [IAM permissions needed for the AWS Glue Studio user \(p. 6\)](#)
- [Overview of Job-related permissions \(p. 8\)](#)
- [Set up IAM permissions for AWS Glue Studio \(p. 9\)](#)
- [Configuring a VPC for your ETL job \(p. 9\)](#)
- [Populate the AWS Glue Data Catalog \(p. 10\)](#)

Sign up for AWS

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Create an IAM administrator user

If your account already includes an IAM user with full AWS administrative permissions, you can skip this section.

To create an administrator user for yourself and add the user to an administrators group (console)

1. Sign in to the [IAM console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

Note

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user that follows and securely lock away the root user credentials. Sign in as the root user only to perform a few [account and service management tasks](#).

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **Administrator**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.

5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **Administrators**.
10. Choose **Filter policies**, and then select **AWS managed - job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

Note

You must activate IAM user and role access to Billing before you can use the `AdministratorAccess` permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in [step 1 of the tutorial about delegating access to the billing console](#).

12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see [Tagging IAM entities](#) in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see [Access management](#) and [Example policies](#).

Signing in as an IAM user

Sign in to the [IAM console](#) by choosing **IAM user** and entering your AWS account ID or account alias. On the next page, enter your IAM user name and your password.

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose the sign-in link beneath the button to return to the main sign-in page. From there, you can enter your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

IAM permissions needed for the AWS Glue Studio user

To use AWS Glue Studio, the user must have access to various AWS resources. The user must be able to view and select Amazon S3 buckets, IAM policies and roles, and AWS Glue Data Catalog objects.

AWS Glue service permissions

AWS Glue Studio uses the actions and resources of the AWS Glue service. Your user needs permissions on these actions and resources to effectively use AWS Glue Studio. You can grant the AWS Glue Studio

user the `AWSGlueConsoleFullAccess` managed policy, or create a custom policy with a smaller set of permissions.

Important

Per security best practices, it is recommended to restrict access by tightening policies to further restrict access to Amazon S3 bucket and Amazon CloudWatch log groups. For an example Amazon S3 policy, see [Writing IAM Policies: How to Grant Access to an Amazon S3 Bucket](#).

Data preview permissions

Data previews enable you to see a sample of your data at any stage of your job (reading, transforming, writing), without having to run the job. You specify an AWS Identity and Access Management (IAM) role for AWS Glue Studio to use when accessing the data. IAM roles are intended to be assumable and do not have standard long-term credentials such as a password or access keys associated with it. Instead, when AWS Glue Studio assumes the role, IAM provides it with temporary security credentials.

To ensure data previews work correctly, you must configure a role trust policy for the role in IAM. Add the AWS Glue service as a principal in this trust policy.

To configure a role trust policy for using data previews with AWS Glue Studio

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the left-side navigation, choose **Roles**.
3. Locate the role used for data previews or your ETL job, and then choose the role name.
4. Choose the **Trust Relationships** tab, and then choose the **Edit trust relationship** button.
5. Copy and paste the following block in the **Policy Document** text field.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Choose **Update Trust Policy** to complete the configuration.

Amazon CloudWatch permissions

You can monitor your AWS Glue Studio jobs using Amazon CloudWatch, which collects and processes raw data from AWS Glue into readable, near-real-time metrics. By default, AWS Glue metrics data is sent to CloudWatch automatically. For more information, see [What Is Amazon CloudWatch?](#) in the *Amazon CloudWatch User Guide*, and [AWS Glue Metrics](#) in the *AWS Glue Developer Guide*.

To access CloudWatch dashboards, the user accessing AWS Glue Studio needs one of the following:

- The `AdministratorAccess` policy
- The `CloudWatchFullAccess` policy
- A custom policy that includes one or more of these specific permissions:

- `cloudwatch:GetDashboard` and `cloudwatch:ListDashboards` to view dashboards
- `cloudwatch:PutDashboard` to create or modify dashboards
- `cloudwatch:DeleteDashboards` to delete dashboards

For more information for changing permissions for an IAM user using policies, see [Changing Permissions for an IAM User](#) in the *IAM User Guide*.

Overview of Job-related permissions

When you create a job using AWS Glue Studio, the job assumes the permissions of the IAM role that you specify when you create it. This IAM role must have permission to extract data from your data source, write data to your target, and access AWS Glue resources.

The name of the role that you create for the job must start with the string `AWSGlueServiceRole` for it to be used correctly by AWS Glue Studio. For example, you might name your role `AWSGlueServiceRole-FlightDataJob`.

Data source and data target permissions

An AWS Glue Studio job must have access to Amazon S3 for any sources, targets, scripts, and temporary directories that you use in your job. You can create a policy to provide fine-grained access to specific Amazon S3 resources.

- Data sources require `s3:ListBucket` and `s3:GetObject` permissions.
- Data targets require `s3:ListBucket`, `s3:PutObject`, and `s3>DeleteObject` permissions.

If you choose Amazon Redshift as your data source, you can provide a role for cluster permissions. Jobs that run against a Amazon Redshift cluster issue commands that access Amazon S3 for temporary storage using temporary credentials. If your job runs for more than an hour, these credentials will expire causing the job to fail. To avoid this problem, you can assign a role to the Amazon Redshift cluster itself that grants the necessary permissions to jobs using temporary credentials. For more information, see [Moving Data to and from Amazon Redshift](#) in the *AWS Glue Developer Guide*.

If the job uses data sources or targets other than Amazon S3, then you must attach the necessary permissions to the IAM role used by the job to access these data sources and targets. For more information, see [Setting Up Your Environment to Access Data Stores](#) in the *AWS Glue Developer Guide*.

If you're using connectors and connections for your data store, you need additional permissions, as described in [the section called "Permissions required for using connectors"](#) (p. 9).

Permissions required for deleting jobs

In AWS Glue Studio you can select multiple jobs in the console to delete. To perform this action, you must have the `glue:BatchDeleteJob` permission. This is different from the AWS Glue console, which requires the `glue:DeleteJob` permission for deleting jobs.

AWS Key Management Service permissions

If you plan to access Amazon S3 sources and targets that use server-side encryption with AWS Key Management Service (AWS KMS), then attach a policy to the AWS Glue Studio role used by the job that enables the job to decrypt the data. The job role needs the `kms:ReEncrypt`, `kms:GenerateDataKey`, and `kms:DescribeKey` permissions. Additionally, the job role needs the `kms:Decrypt` permission

to upload or download an Amazon S3 object that is encrypted with an AWS KMS customer master key (CMK).

There are additional charges for using AWS KMS CMKs. For more information, see [AWS Key Management Service Concepts - Customer Master Keys \(CMKs\)](#) and [AWS Key Management Service Pricing](#) in the *AWS Key Management Service Developer Guide*.

Permissions required for using connectors

If you're using an AWS Glue Custom Connector and connection to access a data store, the role used to run the AWS Glue ETL job needs additional permissions attached:

- The AWS managed policy `AmazonEC2ContainerRegistryReadOnly` for accessing connectors purchased from AWS Marketplace.
- The `glue:GetJob` and `glue:GetJobs` permissions.
- AWS Secrets Manager permissions for accessing secrets that are used with connections. Refer to [IAM policy examples for secrets in AWS Secrets Manager](#) for example IAM policies.

If your AWS Glue ETL job runs within a VPC running Amazon VPC, then the VPC must be configured as described in [the section called "Configuring a VPC for your ETL job"](#) (p. 9).

Set up IAM permissions for AWS Glue Studio

You can create the roles and assign policies to users and job roles by using the AWS administrator user.

To create an IAM policy and role for use with AWS Glue Studio

1. Create an IAM policy for the AWS Glue service.

You can use the **AWSGlueConsoleFullAccess** AWS managed policy.

To create your own policy, follow the steps documented in [Create an IAM Policy for the AWS Glue Service](#) in the *AWS Glue Developer Guide*.

2. Create an IAM role for AWS Glue and attach the IAM policy to this role.

Follow the steps documented in [Create an IAM Role for AWS Glue](#) in the *AWS Glue Developer Guide*.

3. Create a user for AWS Glue or AWS Glue Studio.

You can either use the administrator user for configuring AWS Glue resources, or you can create a separate user for accessing AWS Glue Studio.

To create additional users for AWS Glue and AWS Glue Studio, follow the steps in [Creating Your First IAM Delegated User and Group](#) in the *IAM User Guide*.

Configuring a VPC for your ETL job

You can use Amazon Virtual Private Cloud (Amazon VPC) to define a virtual network in your own logically isolated area within the AWS Cloud, known as a *virtual private cloud (VPC)*. You can launch your AWS resources, such as instances, into your VPC. Your VPC closely resembles a traditional network that you might operate in your own data center, with the benefits of using the scalable infrastructure of AWS. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables, network gateways, and security settings. You can connect instances in your VPC to the internet.

You can connect your VPC to your own corporate data center, making the AWS Cloud an extension of your data center. To protect the resources in each subnet, you can use multiple layers of security, including security groups and network access control lists. For more information, see the [Amazon VPC User Guide](#).

You can configure your AWS Glue ETL jobs to run within a VPC when using connectors. You must configure your VPC for the following, as needed:

- Public network access for data stores not in AWS. All data stores that are accessed by the job must be available from the VPC subnet.
- If your job needs to access both VPC resources and the public internet, the VPC needs to have a network address translation (NAT) gateway inside the VPC.

For more information, see [Setting Up Your Environment to Access Data Stores](#) in the *AWS Glue Developer Guide*.

Populate the AWS Glue Data Catalog

AWS Glue Studio uses datasets that are defined in the AWS Glue Data Catalog. These datasets are used as sources and targets for ETL workflows in AWS Glue Studio. If you choose the Data Catalog for your data source or target, then the Data Catalog tables related to your data source or data target must exist prior to creating a job.

When reading from or writing to a data source, your ETL job needs to know the schema of the data. The ETL job can get this information from a table in the AWS Glue Data Catalog. You can use a crawler, the AWS Glue console, AWS CLI, or an AWS CloudFormation template file to add databases and tables to the Data Catalog. For more information about populating the Data Catalog, see [Data Catalog](#) in the *AWS Glue Developer Guide*.

When using connectors, you can use the schema builder to enter the schema information when you configure the data source node of your ETL job in AWS Glue Studio. For more information, see [the section called "Authoring jobs with custom connectors" \(p. 50\)](#).

If you choose an Amazon S3 location as your data source, AWS Glue Studio can automatically infer the schema of the data it reads from the files at the specified location. For more information, see [Using files in Amazon S3 for the data source \(p. 22\)](#).

If you choose a streaming data source, AWS Glue Studio can automatically infer the schema of the data it reads from the data stream. For more information, see [Using a streaming data source \(p. 23\)](#).

Tutorial: Getting started with AWS Glue Studio

You can use AWS Glue Studio to create jobs that extract structured or semi-structured data from a data source, perform a transformation of that data, and save the result set in a data target.

Topics

- [Prerequisites \(p. 11\)](#)
- [Step 1: Start the job creation process \(p. 11\)](#)
- [Step 2: Edit the data source node in the job diagram \(p. 12\)](#)
- [Step 3: Edit the transform node of the job \(p. 13\)](#)
- [Step 4: Edit the data target node of the job \(p. 13\)](#)
- [Step 5: View the job script \(p. 14\)](#)
- [Step 6: Specify the job details and save the job \(p. 14\)](#)
- [Step 7: Run the job \(p. 15\)](#)
- [Next steps \(p. 15\)](#)

Prerequisites

This tutorial has the following prerequisites:

- You have an AWS account.
- You have access to AWS Glue Studio.
- Your account has all the necessary permissions for creating and running a job for an Amazon S3 data source and data target.
- You have created an AWS Identity and Access Management role for the job to use. You can also choose an IAM role for the job that includes permissions for all your data sources, data targets, temporary directory, scripts, and any libraries used by the job.
- The following components exist in AWS:
 - The `Flights Data Crawler` crawler
 - The `flights-db` database
 - The `flightscsv` table
 - The IAM role `AWSGlueServiceRole-CrawlerTutorial`

To create these components, you can complete the service tutorial **Add a crawler**, which populates the AWS Glue Data Catalog with the necessary objects. This tutorial also creates an IAM role with the necessary permissions. You can find the tutorial on the AWS Glue service page at <https://console.aws.amazon.com/glue/>. The tutorial is located in the left-side navigation, under **Tutorials**. Alternatively, you can use the documentation version of this tutorial, [Tutorial: Adding an AWS Glue crawler \(p. 79\)](#).

Step 1: Start the job creation process

In this task, you choose to start the job creation by using a template.

To create a job, starting with a template

1. Sign in to the AWS Management Console and open the AWS Glue Studio console at <https://console.aws.amazon.com/gluestudio/>.
2. On the AWS Glue Studio landing page, choose **View jobs** under the heading **Create and manage jobs**.
3. On the **Jobs** page, under the heading **Create job**, choose the **Source and target added to the graph** option. Then, choose **S3** for the **Source** and **S3** for the **Target**.
4. Choose the **Create** button to start the job creation process.

The job editing page opens with a simple three-node job diagram displayed.

Step 2: Edit the data source node in the job diagram

Choose the **Data source - S3 bucket** node in the job diagram to edit the data source properties.

To edit the data source node

1. On the **Node properties** tab in the node details pane, for **Name**, enter a name that is unique for this job.

The value you enter is used as the label for the data source node in the job diagram. If you use unique names for the nodes in your job, then it's easier to identify each node in the job diagram, and also to select parent nodes. For this tutorial, enter the name **S3 Flight Data**.
2. Choose the **Data source properties - S3** tab in the node details panel.
3. Choose the **Data Catalog table** option for the S3 source type.
4. For **Database**, choose the **flights-db** database from the list of available databases in your AWS Glue Data Catalog.
5. For **Table**, enter **flight** in the search field, and then choose the **flightscsv** table from your AWS Glue Data Catalog.
6. (Optional) Choose the **Output schema** tab in the node details panel to view the data schema.
7. (Optional) After configuring the node properties and data source properties, you can preview the dataset from your data source by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

By default, the first 5 columns are selected for viewing in the **Data preview** tab. To view other columns, choose **Previewing 5 of 65 fields**. For example, you can deselect the first 5 columns and select **fl_date**, **airline_id**, **fl_num**, **tail_num**, and **origin_airport_id**. Scroll to the end of the column list and choose **Confirm** to save your choices.

After you have provided the required information for the data source node, a green check mark appears on the node in the job diagram.

Step 3: Edit the transform node of the job

The transform node is where you specify how you want to modify the data from its original format. An *ApplyMapping* transform enables you to rename data property keys, change the data types, and drop columns from the dataset.

When you edit the **Transform - ApplyMapping** node, the original schema for your data is shown in the **Source key** column in the node details panel. This is the data property key name (column name) that is obtained from the source data and stored in the table in the AWS Glue Data Catalog.

The **Target key** column shows the key name that will appear in the data target. You can use this field to change the data property key name in the output. The **Data type** column shows the data type of the key and allows you to change it to different data type for the target. The **Drop** column contains a check box. This box allows you to choose a field to drop it from the target schema.

To edit the transform node

1. Choose the **Transform - ApplyMapping** node in the job diagram to edit the data transformation properties.
2. In the node details panel, on the **Node properties** tab, review the information.

You can change the name of this node if you want.

3. Choose the **Transform** tab in the node details panel.
4. Choose to drop the keys `quarter` and `day_of_week` by selecting the check box in the **Drop** column for each key.
5. For the key that shows `day_of_month` in the **Source key** column, change the **Target key** value to `day`.

Change the data type for the `month` and `day` keys to **tinyint**. The **tinyint** data type stores integers using 1 byte of storage, with a range of values from 0 to 255. When changing the data type, you must verify that the data type is supported by your target.

6. (Optional) Choose the **Output schema** tab in the node details panel to view the modified schema.
7. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

By default, the first 5 columns are selected for the data preview, but the columns are no longer the same as the columns viewed on the data source node because we dropped two of the columns and renamed a third column.

Notice that the **Transform - Apply Mapping** node in the job diagram now has a green check mark, indicating that the node has been edited and has all the required information.

Step 4: Edit the data target node of the job

A data target node determines where the transformed output is sent. The location can be an Amazon S3 bucket, a Data Catalog table, or a connector and connection. If you choose a Data Catalog table, the data is written to the location associated with that table. For example, if you use a crawler to create a table in the Data Catalog for a JDBC target, the data is written to that JDBC table.

To edit the data target node

1. Choose the **Data target - S3 bucket** node in the job diagram to edit the data target properties.
2. In the node details panel on the right, choose the **Node properties** tab. For **Name**, enter a unique name for the node, such as **Revised Flight Data**.
3. Choose the **Data target properties - S3** tab.
4. For **Format**, choose **JSON**.

For **Compression Type**, keep the default value of **None**.

For the **S3 Target Location**, choose the **Browse S3** button to see the Amazon S3 buckets that you have access to, and choose one as the target destination.

For the **Data Catalog update options**, keep the default setting of **Do not update the Data Catalog**.

For more information about the available options, see [Overview of data target options \(p. 38\)](#).

Step 5: View the job script

After you configure all the nodes in the job, AWS Glue Studio generates a script that is used by the job to read, transform, and write the data in the target location.

To view the script, choose the **Script** tab at the top of the job editing pane. Don't click the **Edit script** button, because this will take you out of visual editor mode.

If you clicked the **Edit script** button and confirmed your choice, you can reload the page (without saving the job first), to reset the **Script** tab.

Step 6: Specify the job details and save the job

Before you can save and run your extract, transform, and load (ETL) job, you must first enter additional information about the job itself.

To specify the job details and save the job

1. Choose the **Job details** tab.
2. Enter a name for the job—for example **FlightDataETL**. Provide a UTF-8 string with a maximum length of 255 characters.

You can optionally enter a description of the job.

3. For the **IAM role**, choose **AWSGlueServiceRole-CrawlerTutorial** from the list of available roles. You might have to add access to the target Amazon S3 bucket to this role.

If you have many roles to choose from, you can start entering part of the role name in the **IAM role** search field, and the roles with the matching text string will be displayed. For example, you can enter **tutorial** in the search field to find all roles with **tutorial** (case-insensitive) in the name.

The AWS Identity and Access Management (IAM) role is used to authorize access to resources that are used to run the job. You can only choose roles that already exist in your account. The role you choose must have permission to access your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job, as well as access to AWS Glue service resources.

For the steps to create a role, see [Create an IAM Role for AWS Glue](#) in the *AWS Glue Developer Guide*.

4. For the remaining fields, use the default values.

5. Choose **Save** in the top-right corner of the page.

You should see a notification at the top of the page that the job was successfully saved.

If you don't see a notification that your job was successfully saved, then there is most likely information missing that prevents the job from being saved.

- Review the job in the visual editor, and choose any node that doesn't have a green check mark.
- If any of the tabs above the visual editor pane have a callout, choose that tab and look for any fields that are highlighted in red.

Step 7: Run the job

Now that the job has been saved, you can run the job. Choose the **Run** button at the top of the page. You should then see a notification that the job was successfully started.

You can choose either the link in the notification for **Run Details**, or choose the **Runs** tab to view the run status of the job.

On the **Runs** tab, there is a card for each recent run of the job with information about that job run.

For more information about the job run information, see [the section called “View information for recent job runs” \(p. 68\)](#).

Next steps

After you start the job run, you might want to try some of the following tasks:

- View the job monitoring dashboard – [Accessing the job monitoring dashboard \(p. 74\)](#).
- Try a different transform on the data – [Overview of mappings and transforms \(p. 26\)](#).
- View the jobs that exist in your account – [View your jobs \(p. 68\)](#).
- Run the job using a time-based schedule – [Schedule job runs \(p. 66\)](#).

Creating ETL jobs with AWS Glue Studio

You can use the simple visual interface in AWS Glue Studio to create your ETL jobs. You use the **Jobs** page to create new jobs. You can also use a script editor to work directly with code in the AWS Glue Studio ETL job script.

On the **Jobs** page, you can see all the jobs that you have created either with AWS Glue Studio or AWS Glue. You can view, manage, and run your jobs on this page.

Topics

- [Start the job creation process \(p. 16\)](#)
- [Create jobs that use a connector \(p. 17\)](#)
- [Next steps for creating a job in AWS Glue Studio \(p. 17\)](#)

Start the job creation process

You use the visual editor to create and customize your jobs. When you create a new job, you have the option of starting with an empty canvas, a job with a data source, transform, and data target node, or writing an ETL script.

To create a job in AWS Glue Studio

1. Sign in to the AWS Management Console and open the AWS Glue Studio console at <https://console.aws.amazon.com/gluestudio/>.
2. You can either choose **Create and manage jobs** from the AWS Glue Studio landing page, or you can choose **Jobs** from the navigation pane.

The **Jobs** page appears.

3. In the **Create job** section, choose a configuration option for your job.
 - To create a job starting with an empty canvas, choose **Visual with a blank canvas**.
 - To create a job starting with source node, or with a source, transform and target node, choose **Visual with a source and target**.

You then choose the data source type. You can also choose the data target type, or you can choose the **Choose later** option to start with only a data source node in the graph.

- For those familiar with programming and writing ETL scripts, you can choose **Spark script editor** to create a new Spark ETL job. You then have the option of writing Python or Scala code in a script editor window, or uploading an existing script from a local file. If you choose to use the script editor, then you can't use the visual job editor.

By default, new scripts are coded in Python. To write a new Scala script, see [Creating and editing Scala scripts in AWS Glue Studio \(p. 42\)](#).

- You can alternatively create a new Python shell job with the **Python Shell script editor** option. You then have the option of writing code in a script editor window starting with a template (boilerplate), or uploading an existing script from a local file. If you choose to use the Python shell editor, then you can't use the visual job editor.

4. Choose **Create** to open the visual job editor.

The screenshot shows the 'Create job' interface in AWS Glue Studio. The 'Visual with a source and target' option is selected. The 'Source' dropdown menu is open, showing a list of data sources: Amazon S3 (selected), AWS Glue Data Catalog, Amazon S3, Amazon Kinesis, Apache Kafka, Relational DB, Amazon Redshift, MySQL, and PostgreSQL. The 'Target' dropdown menu is also open, showing 'Amazon S3'. To the right, there is a table of recent job runs.

Type	Last modified
Glue ETL	8/12/2021, 4:51:48 PM
Glue ETL	8/12/2021, 1:54:17 PM
Glue ETL	8/12/2021, 1:51:28 PM
Glue ETL	6/4/2021, 8:44:32 PM
Glue ETL	6/4/2021, 8:44:17 PM
Glue ETL	5/21/2021, 6:32:35 PM

Create jobs that use a connector

After you have added a connector to AWS Glue Studio and created a connection for that connector, you can create a job that uses the connection for the data source.

For detailed instructions, see [the section called “Authoring jobs with custom connectors” \(p. 50\)](#).

Next steps for creating a job in AWS Glue Studio

You use the visual job editor to configure nodes for your job. Each node represents an action, such as reading data from the source location or applying a transform to the data. Each node you add to your job has properties that provide information about either the data location or the transform.

The next steps for creating and managing your jobs are:

- [Editing ETL jobs in AWS Glue Studio \(p. 18\)](#)
- [Adding connectors to AWS Glue Studio \(p. 47\)](#)
- [View the job script \(p. 69\)](#)
- [Modify the job properties \(p. 69\)](#)
- [Save the job \(p. 71\)](#)
- [Start a job run \(p. 66\)](#)
- [View information for recent job runs \(p. 68\)](#)
- [Accessing the job monitoring dashboard \(p. 74\)](#)

Editing ETL jobs in AWS Glue Studio

While creating a new job, or after you have saved your job, you can use can AWS Glue Studio to modify your ETL jobs. You can do this by editing the nodes in the visual editor or by editing the job script in developer mode. You can also add and remove nodes in the visual editor to create more complicated ETL jobs.

Topics

- [Accessing the job diagram editor \(p. 18\)](#)
- [Job editor features \(p. 18\)](#)
- [Editing the data source node \(p. 20\)](#)
- [Editing the data transform node \(p. 25\)](#)
- [Configuring data target nodes \(p. 38\)](#)
- [Editing or uploading a job script \(p. 41\)](#)
- [Adding nodes to the job diagram \(p. 44\)](#)
- [Changing the parent nodes for a node in the job diagram \(p. 44\)](#)
- [Deleting nodes from the job diagram \(p. 45\)](#)

Accessing the job diagram editor

Use the AWS Glue Studio job editor to edit your ETL jobs.

You can access the job editor in the following ways:

- Choose **Jobs** in the console navigation pane. On the **Jobs** page, locate the job in the **Your jobs** list. You can then either:
 - Choose the name of the job in the **Name** column to open the job editor for that job.
 - Choose the job, and then choose **Edit job** from the **Actions** list.
- Choose **Monitoring** in the console navigation pane. On the **Monitoring** page, locate the job in the **Job runs** list. You can filter the rows in the **Job runs** list, as described in [Job runs view \(p. 74\)](#). Choose the job you want to edit, and then choose **View job** from the **Actions** menu.

Job editor features

The job editor provides the following features for creating and editing jobs.

- A visual diagram of your job, with a node for each job task: Data source nodes for reading the data; transform nodes for modifying the data; data target nodes for writing the data.

You can view and configure the properties of each node in the job diagram. You can also view the schema and sample data for each node in the job diagram. These features help you to verify that your job is modifying and transforming the data in the right way, without having to run the job.

- A Script viewing and editing tab, where you can modify the code generated for your job.
- A Job details tab, where you can configure a variety of settings to customize the environment in which your AWS Glue ETL job runs.
- A Runs tab, where you can view the current and previous runs of the job, view the status of the job run, and access the logs for the job run.
- A Schedules tab, where you can configure the start time for your job, or set up a recurring job runs.

Using schema previews in the visual job editor

While creating or editing your job, you can use the **Output schema** tab to view the schema for your data.

Before you can see the schema, the job editor needs permissions to access the data source. You can specify an IAM role on the Job details tab of the editor or on the **Output schema** tab for a node. If the IAM role has all the necessary permissions to access the data source, you can then view the schema on the **Output schema** tab for a node.

Using data previews in the visual job editor

While creating or editing your job, you can use the **Data preview** tab to view a sample of your data.

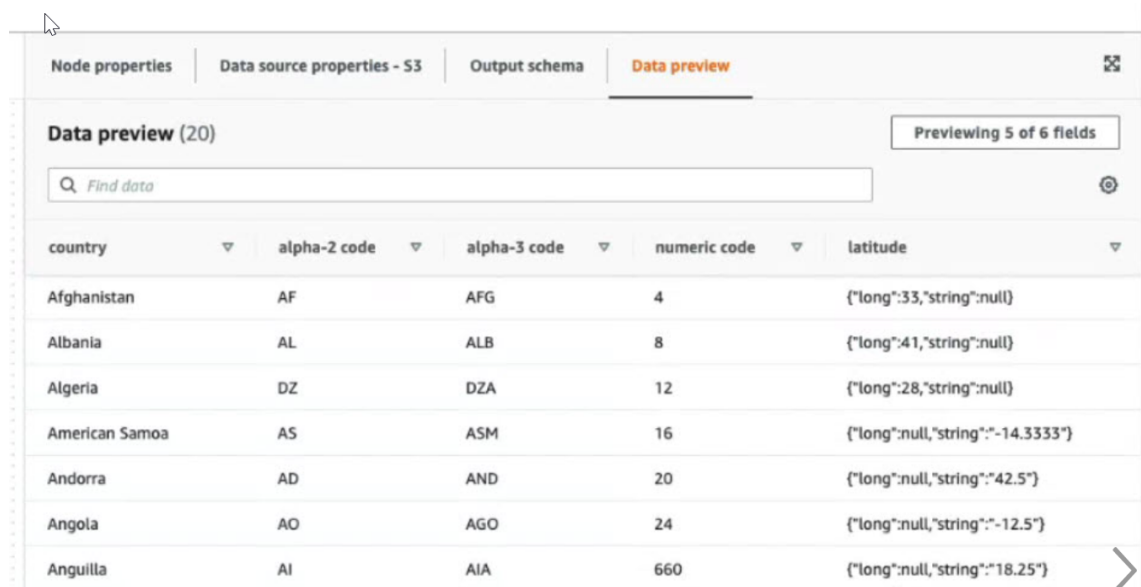
Before you can see the data sample, the job editor needs permissions to access the data source. The first time you choose the **Data preview** tab, you are prompted to choose an IAM role to use. This can be the same role that you plan to use for your job, or it can be a different role. The IAM role you choose must have the necessary permissions to create the data previews.

After you choose an IAM role, it takes about 20 to 30 seconds before the data appears. You are charged for data preview usage as soon as you choose the IAM role. The following features help you when viewing the data.

- Choose the settings icon (a gear symbol) to configure your preferences for data previews. You can change the sample size or you can choose to wrap the text from one line to the next. These settings apply to all nodes in the job diagram.
- Choose the **Previewing x of y fields** button to select which columns (fields) to preview. When you preview your data using the default settings, the job editor shows the first 5 columns of your dataset. You can change this to show all or none (not recommended).
- You can scroll through the data preview window both horizontally and vertically.
- Use the split/whole screen button to expand the Data preview tab to the entire screen (over-laying the job graph), to better view the data and data structures.

Data previews help you create and test your job, without having to repeatedly run the job.

- You can test an IAM role to make sure you have access to your data sources or data targets.
- You can check that the transform is modifying the data in the intended way. For example, if you use a Filter transform, you can make sure that the filter is selecting the right subset of data.
- If your dataset contains columns with values of multiple types, the data preview shows a list of tuples for these columns. Each tuple contains the data type and its value, as shown in the following screenshot.



The screenshot shows the 'Data preview' tab in AWS Glue Studio. The tab is titled 'Data preview (20)' and indicates 'Previewing 5 of 6 fields'. A search bar with the placeholder 'Find data' is present. Below the search bar is a table with the following columns: country, alpha-2 code, alpha-3 code, numeric code, and latitude. The table displays data for several countries, including Afghanistan, Albania, Algeria, American Samoa, Andorra, Angola, and Anguilla. The 'latitude' column contains JSON-like strings representing coordinates.

country	alpha-2 code	alpha-3 code	numeric code	latitude
Afghanistan	AF	AFG	4	{"long":33,"string":null}
Albania	AL	ALB	8	{"long":41,"string":null}
Algeria	DZ	DZA	12	{"long":28,"string":null}
American Samoa	AS	ASM	16	{"long":null,"string":"-14.3333"}
Andorra	AD	AND	20	{"long":null,"string":"42.5"}
Angola	AO	AGO	24	{"long":null,"string":"-12.5"}
Anguilla	AI	AIA	660	{"long":null,"string":"18.25"}

Restrictions when using data previews

When using data previews, you might encounter the following restrictions or limitations.

- The first time you choose the Data preview tab you must choose IAM role. This role must have the necessary permissions to access the data and other resources needed to create the data previews.
- After you provide an IAM role, it takes a while before the data is available for viewing. For datasets with less than 1 GB of data, it can take up to one minute. If you have a large dataset, you should use partitions to improve the loading time. Loading data directly from Amazon S3 has the best performance.
- If you have a very large dataset, and it takes more than 30 minutes to query the data for the data preview, the request will time out. You can reduce the dataset size to use data previews.
- By default, you see the first 5 columns in the Data preview tab. If the columns have no data values, you will get a message that there is no data to display. You can increase the number of rows sampled, or selected different columns to see data values.
- Data previews are currently not supported for streaming data sources, or for data sources that use custom connectors.
- Errors on one node effect the entire job. If any one node has an error with data previews, the error will show up on all nodes until you correct it.
- If you change a data source for the job, then the child nodes of that data source might need to be updated to match the new schema. For example, if you have an ApplyMapping node that modifies a column, and the column does not exist in the replacement data source, you will need to update the ApplyMapping transform node.
- If you view the Data preview tab for a SQL query transform node, and the SQL query uses an incorrect field name, the Data preview tab shows an error.

Editing the data source node

To specify the data source properties, you first choose a data source node in the job diagram. Then, on the right side in the node details panel, you configure the node properties.

To modify the properties of a data source node

1. Go to the visual editor for a new or saved job.
2. Choose a data source node in the job diagram.
3. Choose the **Node properties** tab in the node details panel, and then enter the following information:
 - **Name:** (Optional) Enter a name to associate with the node in the job diagram. This name should be unique among all the nodes for this job.
 - **Node type:** The node type determines the action that is performed by the node. In the list of options for **Node type**, choose one of the values listed under the heading **Data source**.
4. Configure the **Data source properties** information. For more information, see the following sections:
 - [Using Data Catalog tables for the data source](#) (p. 21)
 - [Using a connector for the data source](#) (p. 22)
 - [Using files in Amazon S3 for the data source](#) (p. 22)
 - [Using a streaming data source](#) (p. 23)
5. (Optional) After configuring the node properties and data source properties, you can view the schema for your data source by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
6. (Optional) After configuring the node properties and data source properties, you can preview the dataset from your data source by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Using Data Catalog tables for the data source

For all data sources except Amazon S3 and connectors, a table must exist in the AWS Glue Data Catalog for the source type that you choose. AWS Glue Studio does not create the Data Catalog table.

To configure a data source node based on a Data Catalog table

1. Go to the visual editor for a new or saved job.
2. Choose a data source node in the job diagram.
3. Choose the **Data source properties** tab, and then enter the following information:
 - **S3 source type:** (For Amazon S3 data sources only) Choose the option **Select a Catalog table** to use an existing AWS Glue Data Catalog table.
 - **Database:** Choose the database in the Data Catalog that contains the source table you want to use for this job. You can use the search field to search for a database by its name.
 - **Table:** Choose the table associated with the source data from the list. This table must already exist in the AWS Glue Data Catalog. You can use the search field to search for a table by its name.
 - **Partition predicate:** (For Amazon S3 data sources only) Enter a Boolean expression based on Spark SQL that includes only the partitioning columns. For example: `"(year== '2020' and month== '04')"`
 - **Temporary directory:** (For Amazon Redshift data sources only) Enter a path for the location of a working directory in Amazon S3 where your ETL job can write temporary intermediate results.
 - **Role associated with the cluster:** (For Amazon Redshift data sources only) Enter a role for your ETL job to use that contains permissions for Amazon Redshift clusters. For more information, see [the section called "Data source and data target permissions"](#) (p. 8).

Using a connector for the data source

If you select a connector for the **Node type**, follow the instructions at [Authoring jobs with custom connectors \(p. 50\)](#) to finish configuring the data source properties.

Using files in Amazon S3 for the data source

If you choose Amazon S3 as your data source, then you can choose either:

- A Data Catalog database and table.
- A bucket, folder, or file in Amazon S3.

If you use an Amazon S3 bucket as your data source, AWS Glue Studio detects the schema of the data at the specified location from one of the files, or by using the file you specify as a sample file. Schema detection occurs when you use the **Infer schema** button. If you change the Amazon S3 location or the sample file, then you must choose **Infer schema** again to perform the schema detection using the new information.

To configure a data source node that reads directly from files in Amazon S3

1. Go to the visual editor for a new or saved job.
2. Choose a data source node in the job diagram for an Amazon S3 source.
3. Choose the **Data source properties** tab, and then enter the following information:
 - **S3 source type:** (For Amazon S3 data sources only) Choose the option **S3 location**.
 - **S3 URL:** Enter the path to the Amazon S3 bucket, folder, or file that contains the data for your job. You can choose **Browse S3** to select the path from the locations available to your account.
 - **Recursive:** Choose this option if you want AWS Glue Studio to read data from files in child folders at the S3 location.

If the child folders contain partitioned data, AWS Glue Studio doesn't add any partition information that's specified in the folder names to the Data Catalog. For example, consider the following folders in Amazon S3:

```
S3://sales/year=2019/month=Jan/day=1  
S3://sales/year=2019/month=Jan/day=2
```

If you choose **Recursive** and select the `sales` folder as your S3 location, then AWS Glue Studio reads the data in all the child folders, but doesn't create partitions for year, month or day.

- **Data format:** Choose the format that the data is stored in. You can choose JSON, CSV, or Parquet. The value you select tells the AWS Glue job how to read the data from the source file.

Note

If you don't select the correct format for your data, AWS Glue Studio might infer the schema correctly, but the job won't be able to correctly parse the data from the source file.

You can enter additional configuration options, depending on the format you choose.

- **JSON** (JavaScript Object Notation)
 - **JsonPath:** Enter a JSON path that points to an object that is used to define a table schema. JSON path expressions always refer to a JSON structure in the same way as XPath expression are used in combination with an XML document. The "root member object" in the JSON path is always referred to as \$, even if it's an object or array. The JSON path can be written in dot notation or bracket notation.

For more information about the JSON path, see [JsonPath](#) on the GitHub website.

- **Records in source files can span multiple lines:** Choose this option if a single record can span multiple lines in the CSV file.
- **CSV** (comma-separated values)
 - **Delimiter:** Enter a character to denote what separates each column entry in the row, for example, `;` or `,`.
 - **Escape character:** Enter a character that is used as an escape character. This character indicates that the character that immediately follows the escape character should be taken literally, and should not be interpreted as a delimiter.
 - **Quote character:** Enter the character that is used to group separate strings into a single value. For example, you would choose **Double quote (")** if you have values such as `"This is a single value"` in your CSV file.
 - **Records in source files can span multiple lines:** Choose this option if a single record can span multiple lines in the CSV file.
 - **First line of source file contains column headers:** Choose this option if the first row in the CSV file contains column headers instead of data.
- **Parquet** (Apache Parquet columnar storage)

There are no additional settings to configure for data stored in Parquet format.

- **Partition predicate:** To partition the data that is read from the data source, enter a Boolean expression based on Spark SQL that includes only the partitioning columns. For example: `"(year=='2020' and month=='04')"`
- **Advanced options:** Expand this section if you want AWS Glue Studio to detect the schema of your data based on a specific file.
 - **Schema inference:** Choose the option **Choose a sample file from S3** if you want to use a specific file instead of letting AWS Glue Studio choose a file.
 - **Auto-sampled file:** Enter the path to the file in Amazon S3 to use for inferring the schema.

If you're editing a data source node and change the selected sample file, choose **Reload schema** to detect the schema by using the new sample file.

4. Choose the **Infer schema** button to detect the schema from the sources files in Amazon S3. If you change the Amazon S3 location or the sample file, you must choose **Infer schema** again to infer the schema using the new information.

Using a streaming data source

You can create streaming extract, transform, and load (ETL) jobs that run continuously and consume data from streaming sources in Amazon Kinesis Data Streams, Apache Kafka, and Amazon Managed Streaming for Apache Kafka (Amazon MSK).

To configure properties to access a streaming data source

1. Go to the visual graph editor for a new or saved job.
2. Choose a data source node in the graph for Apache Kafka, Amazon MSK, or Kinesis Data Streams.
3. Choose the **Data source properties** tab, and then enter the following information:

Kinesis

- **Kinesis source type:** Choose the option **Stream details** to use direct access to the streaming source or choose **Data Catalog table** to use the information stored there instead.

If you choose **Stream details**, specify the following additional information.

- **Location of data stream:** Choose whether the stream is located within the current user account, or if it is located in a different account.
- **Region:** Choose the AWS Region where the stream exists. This information is used to construct the ARN for accessing the data stream.
- **Stream ARN:** Enter the Amazon Resource Name (ARN) for the Kinesis data stream. If the stream is located within the current account, you can choose the stream name from the drop-down list. You can use the search field to search for a data stream by its name or ARN.
- **Data format:** Choose the format used by the data stream from the list.

AWS Glue Studio automatically detects the schema from the streaming data.

If you choose **Data Catalog table**, specify the following additional information.

- **Database:** (Optional) Choose the database in the AWS Glue Data Catalog that contains the table associated with your streaming data source. You can use the search field to search for a database by its name.
- **Table:** (Optional) Choose the table associated with the source data from the list. This table must already exist in the AWS Glue Data Catalog. You can use the search field to search for a table by its name.
- **Detect schema:** Choose this option to have AWS Glue Studio detect the schema from the streaming data, rather than using the schema information in a Data Catalog table. This option is enabled automatically if you choose the **Stream details** option.
- **Starting position:** By default, the ETL job uses the **Earliest** option, which means it reads data starting with the oldest available record in the stream. You can instead choose **Latest**, which indicates the ETL job should start reading from just after the most recent record in the stream.
- **Window size:** By default, your ETL job processes and writes out data in 100-second windows. This allows data to be processed efficiently and permits aggregations to be performed on data that arrives later than expected. You can modify this window size to increase timeliness or aggregation accuracy.

AWS Glue streaming jobs use checkpoints rather than job bookmarks to track the data that has been read.

- **Connection options:** Expand this section to add key-value pairs to specify additional connection options. For information about what options you can specify here, see ["connectionType": "kinesis"](#) in the *AWS Glue Developer Guide*.

Kafka

- **Apache Kafka source:** Choose the option **Stream details** to use direct access to the streaming source or choose **Data Catalog table** to use the information stored there instead.

If you choose **Data Catalog table**, specify the following additional information.

- **Database:** (Optional) Choose the database in the AWS Glue Data Catalog that contains the table associated with your streaming data source. You can use the search field to search for a database by its name.
- **Table:** (Optional) Choose the table associated with the source data from the list. This table must already exist in the AWS Glue Data Catalog. You can use the search field to search for a table by its name.
- **Detect schema:** Choose this option to have AWS Glue Studio detect the schema from the streaming data, rather than storing the schema information in a Data Catalog table. This option is enabled automatically if you choose the **Stream details** option.

If you choose **Stream details**, specify the following additional information.

- **Connection name:** Choose the AWS Glue connection that contains the access and authentication information for the Kafka data stream. You must use a connection with Kafka streaming data sources. If a connection doesn't exist, you can use the AWS Glue console to create a connection for your Kafka data stream.
- **Topic name:** Enter the name of the topic to read from.
- **Data format:** Choose the format to use when reading data from the Kafka event stream.
- **Starting position:** By default, the ETL job uses the **Earliest** option, which means it reads data starting with the oldest available record in the stream. You can instead choose **Latest**, which indicates the ETL job should start reading from just after the most recent record in the stream.
- **Window size:** By default, your ETL job processes and writes out data in 100-second windows. This allows data to be processed efficiently and permits aggregations to be performed on data that arrives later than expected. You can modify this window size to increase timeliness or aggregation accuracy.

AWS Glue streaming jobs use checkpoints rather than job bookmarks to track the data that has been read.

- **Connection options:** Expand this section to add key-value pairs to specify additional connection options. For information about what options you can specify here, see ["connectionType": "kafka"](#) in the *AWS Glue Developer Guide*.

Note

Data previews are not currently supported for streaming data sources.

Editing the data transform node

AWS Glue Studio provides a set of built-in transforms that you can use to process your data. Your data passes from one node in the job diagram to another in a data structure called a `DynamicFrame`, which is an extension to an Apache Spark SQL `DataFrame`.

In the pre-populated diagram for a job, between the data source and data target nodes is the **Transform - ApplyMapping** node. You can configure this transform node to modify your data, or you can use additional transforms.

Topics

- [Overview of mappings and transforms \(p. 26\)](#)
- [Using ApplyMapping to remap data property keys \(p. 26\)](#)
- [Using SelectFields to remove most data property keys \(p. 27\)](#)
- [Using DropFields to keep most data property keys \(p. 27\)](#)
- [Renaming a field in the dataset \(p. 28\)](#)
- [Using Spigot to sample your dataset \(p. 29\)](#)
- [Joining datasets \(p. 29\)](#)
- [Using SplitFields to split a dataset into two \(p. 31\)](#)
- [Overview of SelectFromCollection transform \(p. 31\)](#)
- [Using SelectFromCollection to choose which dataset to keep \(p. 32\)](#)
- [Filtering keys within a dataset \(p. 32\)](#)
- [Find and fill missing values in a dataset \(p. 33\)](#)
- [Using a SQL query to transform data \(p. 34\)](#)
- [Creating a custom transformation \(p. 35\)](#)

Overview of mappings and transforms

The following built-in transforms are available with AWS Glue Studio:

- **ApplyMapping:** Map data property keys in the data source to data property keys in the data target. You can rename keys, modify the data types for keys, and choose which keys to drop from the dataset.
- **SelectFields:** Choose the data property keys that you want to keep.
- **DropFields:** Choose the data property keys that you want to drop.
- **RenameField:** Rename a single data property key.
- **Spigot:** Write samples of the data to an Amazon S3 bucket.
- **Join:** Join two datasets into one dataset using a comparison phrase on the specified data property keys. You can use inner, outer, left, right, left semi, and left anti joins.
- **SplitFields:** Split data property keys into two `DynamicFrames`. Output is a collection of `DynamicFrames`: one with selected data property keys, and one with the remaining data property keys.
- **SelectFromCollection:** Choose one `DynamicFrame` from a collection of `DynamicFrames`. The output is the selected `DynamicFrame`.
- **Filter:** Split a dataset into two, based on a filter condition.
- **Custom transform:** Enter code into a text entry field to use custom transforms. The output is a collection of `DynamicFrames`.
- **SQL:** Enter SparkSQL code into a text entry field to use a SQL query to transform the data. The output is a single `DynamicFrame`.

Using ApplyMapping to remap data property keys

An *ApplyMapping* transform remaps the source data property keys into the desired configured for the target data. In an *ApplyMapping* transform node, you can:

- Change the name of multiple data property keys.
- Change the data type of the data property keys, if the new data type is supported and there is a transformation path between the two data types.
- Choose a subset of data property keys by indicating which data property keys you want to drop.

You can add additional *ApplyMapping* nodes to the job diagram as needed – for example, to modify additional data sources or following a *Join* transform.

Note

The *ApplyMapping* transform is not case-sensitive.

To add an ApplyMapping transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **ApplyMapping** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent isn't already selected, choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab in the node details panel.
4. Modify the input schema:
 - To rename a data property key, enter the new name of the key in the **Target key** field.

- To change the data type for a data property key, choose the new data type for the key from the **Data type** list.
 - To remove a data property key from the target schema, choose the **Drop** check box for that key.
5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
 6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Using SelectFields to remove most data property keys

You can create a subset of data property keys from the dataset using the *SelectFields* transform. You indicate which data property keys you want to keep and the rest are removed from the dataset.

Note

The *SelectFields* transform is case sensitive. Use *ApplyMapping* if you need a case-insensitive way to select fields.

To add a SelectFields transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **SelectFields** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab in the node details panel.
4. Under the heading **SelectFields**, choose the data property keys in the dataset that you want to keep. Any data property keys not selected are dropped from the dataset.

You can also choose the check box next to the column heading **Field** to automatically choose all the data property keys in the dataset. Then you can deselect individual data property keys to remove them from the dataset.

5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Using DropFields to keep most data property keys

You can create a subset of data property keys from the dataset using the *DropFields* transform. You indicate which data property keys you want to remove from the dataset and the rest of the keys are retained.

Note

The *DropFields* transform is case sensitive. Use *ApplyMapping* if you need a case-insensitive way to select fields.

To add a DropFields transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **DropFields** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab in the node details panel.
4. Under the heading **DropFields**, choose the data property keys to drop from the data source.

You can also choose the check box next to the column heading **Field** to automatically choose all the data property keys in the dataset. Then you can deselect individual data property keys so they are retained in the dataset.

5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Renaming a field in the dataset

You can use the *RenameField* transform to change the name for an individual property key in the dataset.

Note

The *RenameField* transform is case sensitive. Use *ApplyMapping* if you need a case-insensitive transform.

Tip

If you use the *ApplyMapping* transform, you can rename multiple data property keys in the dataset with a single transform.

To add a RenameField transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **RenameField** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab.
4. Under the heading **Data field**, choose a property key from the source data and then enter a new name in the **New field name** field.
5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.

6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Using Spigot to sample your dataset

To test the transformations performed by your job, you might want to get a sample of the data to check that the transformation works as intended. The *Spigot* transform writes a subset of records from the dataset to a JSON file in an Amazon S3 bucket. The data sampling method can be either a specific number of records from the beginning of the file or a probability factor used to pick records.

To add a Spigot transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **Spigot** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab in the node details panel.
4. Enter an Amazon S3 path or choose **Browse S3** to choose a location in Amazon S3. This is the location where the job writes the JSON file that contains the data sample.
5. Enter information for the sampling method. You can specify a value for **Number of records** to write starting from the beginning of the dataset and a **Probability threshold** (entered as a decimal value with a maximum value of 1) of picking any given record.

For example, to write the first 50 records from the dataset, you would set **Number of records** to 50 and **Probability threshold** to 1 (100%).

Joining datasets

The *Join* transform allows you to combine two datasets into one. You specify the key names in the schema of each dataset to compare. The output `DynamicFrame` contains rows where keys meet the join condition. The rows in each dataset that meet the join condition are combined into a single row in the output `DynamicFrame` that contains all the columns found in either dataset.

To add a Join transform node to your job diagram

1. If there is only one data source available, you must add a new data source node to the job diagram. See [Adding nodes to the job diagram \(p. 44\)](#) for details.
2. Choose one of the source nodes for the join. Choose **Transform** in the toolbar at the top of the visual editor, and then choose **Join** to add a new transform to your job diagram.
3. On the **Node properties** tab, enter a name for the node in the job diagram.
4. In the **Node properties** tab, under the heading **Node parents**, add a parent node so that there are two datasets providing inputs for the join. The parent can be a data source node or a transform node.


Note

A join can have only two parent nodes.

5. Choose the **Transform** tab.

If you see a message indicating that there are conflicting key names, you can either:

- Choose **Resolve it** to automatically add an *ApplyMapping* transform node to your job diagram. The *ApplyMapping* node adds a prefix to any keys in the dataset that have the same name as a key in the other dataset. For example, if you use the default value of **right**, then any keys in the right dataset that have the same name as a key in the left dataset will be renamed to *(right)key name*.
 - Manually add a transform node earlier in the job diagram to remove or rename the conflicting keys.
6. Choose the type of join in the **Join type** list.
- **Inner join**: Returns a row with columns from both datasets for every match based on the join condition. Rows that don't satisfy the join condition aren't returned.
 - **Left join**: All rows from the left dataset and only the rows from the right dataset that satisfy the join condition.
 - **Right join**: All rows from the right dataset and only the rows from the left dataset that satisfy the join condition.
 - **Outer join**: All rows from both datasets.
 - **Left semi join**: All rows from the left dataset that have a match in the right dataset based on the join condition.
 - **Left anti join**: All rows in the left dataset that don't have a match in the right dataset based on join condition.
7. On the **Transform** tab, under the heading **Join conditions**, choose **Add condition**. Choose a property key from each dataset to compare. Property keys on the left side of the comparison operator are referred to as the left dataset and property keys on the right are referred to as the right dataset.

For more complex join conditions, you can add additional matching keys by choosing **Add condition** more than once. If you accidentally add a condition, you can choose the delete icon () to remove it.

8. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
9. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

For an example of the join output schema, consider a join between two datasets with the following property keys:

```
Left: {id, dept, hire_date, salary, employment_status}
Right: {id, first_name, last_name, hire_date, title}
```

The join is configured to match on the `id` and `hire_date` keys using the `=` comparison operator.

Because both datasets contain `id` and `hire_date` keys, you chose **Resolve it** to automatically add the prefix **right** to the keys in the right dataset.

The keys in the output schema would be:

```
{id, dept, hire_date, salary, employment_status,
```

```
(right)id, first_name, last_name, (right)hire_date, title}
```

Using SplitFields to split a dataset into two

The *SplitFields* transform allows you to choose some of the data property keys in the input dataset and put them into one dataset and the unselected keys into a separate dataset. The output from this transform is a collection of `DynamicFrames`.

Note

You must use a *SelectFromCollection* transform to convert the collection of `DynamicFrames` into a single `DynamicFrame` before you can send the output to a target location.

The *SplitFields* transform is case sensitive. Add an *ApplyMapping* transform as a parent node if you need case-insensitive property key names.

To add a SplitFields transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **SplitFields** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab.
4. Choose which property keys you want to put into the first dataset. The keys that you do not choose are placed in the second dataset.
5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.
7. Configure a *SelectFromCollection* transform node to process the resulting datasets.

Overview of SelectFromCollection transform

Certain transforms have multiple datasets as their output instead of a single dataset, for example, *SplitFields*. The *SelectFromCollection* transform selects one dataset (`DynamicFrame`) from a collection of datasets (an array of `DynamicFrames`). The output for the transform is the selected `DynamicFrame`.

You must use this transform after you use a transform that creates a collection of `DynamicFrames`, such as:

- Custom code transforms
- *SplitFields*

If you don't add a *SelectFromCollection* transform node to your job diagram after any of these transforms, you will get an error for your job.

The parent node for this transform must be a node that returns a collection of `DynamicFrames`. If you choose a parent for this transform node that returns a single `DynamicFrame`, such as a *Join* transform, your job returns an error.

Similarly, if you use a *SelectFromCollection* node in your job diagram as the parent for a transform that expects a single *DynamicFrame* as input, your job returns an error.

Node parents

Select which node(s) will provide inputs for this one

Select parents

Split Fields
SplitFields - Transform

⚠ Parent node Split Fields outputs a collection, but node Drop Fields does not accept a collection.

Using SelectFromCollection to choose which dataset to keep

Use the *SelectFromCollection* transform to convert a collection of *DynamicFrames* into a single *DynamicFrame*.

To add a SelectFromCollection transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **SelectFromCollection** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab.
4. Under the heading **Frame index**, choose the array index number that corresponds to the *DynamicFrame* you want to select from the collection of *DynamicFrames*.

For example, if the parent node for this transform is a *SplitFields* transform, on the **Output schema** tab of that node you can see the schema for each *DynamicFrame*. If you want to keep the *DynamicFrame* associated with the schema for **Output 2**, you would select **1** for the value of **Frame index**, which is the second value in the list.

Only the *DynamicFrame* that you choose is included in the output.

5. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
6. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Filtering keys within a dataset

Use the *Filter* transform to create a new dataset by filtering records from the input dataset based on a regular expression. Rows that don't satisfy the filter condition are removed from the output.

- For string data types, you can filter rows where the key value matches a specified string.

- For numeric data types, you can filter rows by comparing the key value to a specified value using the comparison operators `<`, `>`, `=`, `!=`, `<=`, and `>=`.

If you specify multiple filter conditions, the results are combined using an **AND** operator by default, but you can choose **OR** instead.

The *Filter* transform is case sensitive. Add an *ApplyMapping* transform as a parent node if you need case-insensitive property key names.

To add a Filter transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **Filter** to add a new transform to your job diagram, if needed.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent isn't already selected, then choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab.
4. Choose either **Global AND** or **Global OR**. This determines how multiple filter conditions are combined. All conditions are combined using either **AND** or **OR** operations. If you have only a single filter conditions, then you can choose either one.
5. Choose the **Add condition** button in the **Filter condition** section to add a filter condition.

In the **Key** field, choose a property key name from the dataset. In the **Operation** field, choose the comparison operator. In the **Value** field, enter the comparison value. Here are some examples of filter conditions:

- `year >= 2018`
- `State matches 'CA*'`

When you filter on string values, make sure that the comparison value uses a regular expression format that matches the script language selected in the job properties (Python or Scala).

6. Add additional filter conditions, as needed.
7. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
8. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Find and fill missing values in a dataset

You can use the *FillMissingValues* transform to locate records in the dataset that have missing values and add a new field with a value determined by imputation. The input data set is used to train the machine learning (ML) model that determines what the missing value should be. If you use incremental data sets, then each incremental set is used as the training data for the ML model, so the results might not be as accurate.

To use a FillMissingValues transform node in your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **FillMissingValues** to add a new transform to your job diagram, if needed.

2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent isn't already selected, choose a node from the **Node parents** list to use as the input source for the transform.
3. Choose the **Transform** tab.
4. For **Data field**, choose the column or field name from the source data that you want to analyze for missing values.
5. (Optional) In the **New field name** field, enter a name for the field added to each record that will hold the estimated replacement value for the analyzed field. If the analyzed field doesn't have a missing value, the value in the analyzed field is copied into the new field.

If you don't specify a name for the new field, the default name is the name of the analyzed column with `_filled` appended. For example, if you enter **Age** for **Data field** and don't specify a value for **New field name**, a new field named **Age_filled** is added to each record.

6. (Optional) After configuring the transform node properties, you can view the modified schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.
7. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Using a SQL query to transform data

You can use a **SQL** transform to write your own transform in the form of a SQL query.

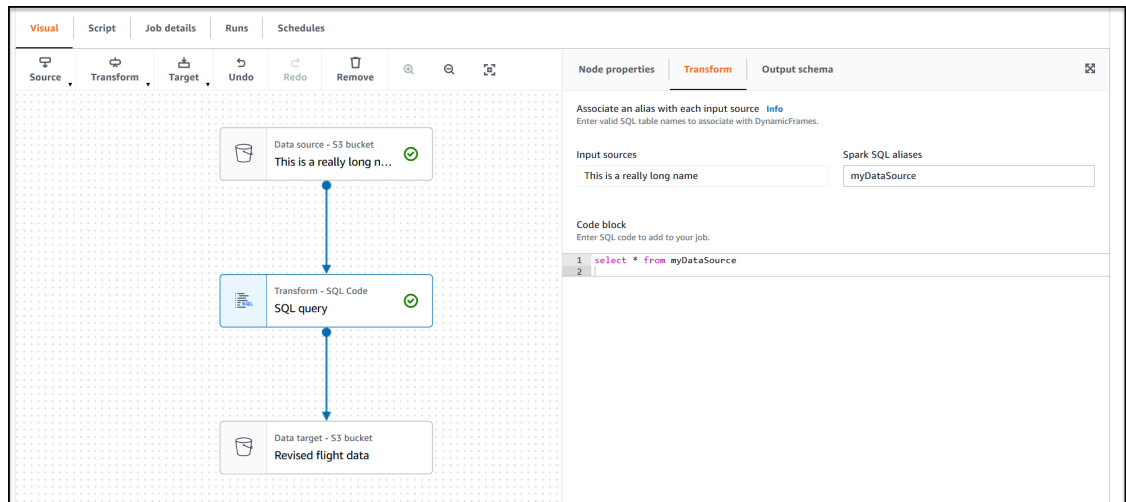
A SQL transform node can have multiple datasets as inputs, but produces only a single dataset as output. It contains a text field, where you enter the Apache SparkSQL query. You can assign aliases to each dataset used as input, to help simplify the SQL query. For more information about the SQL syntax, see the [Spark SQL documentation](#).

Note

If you use a Spark SQL transform with a data source located in a VPC, add an AWS Glue VPC endpoint to the VPC that contains the data source. For more information about configuring development endpoints, see [Adding a Development Endpoint](#), [Setting Up Your Environment for Development Endpoints](#), and [Accessing Your Development Endpoint](#) in the *AWS Glue Developer Guide*.

To use a SQL transform node in your job diagram



1. (Optional) Add a transform node to the job diagram, if needed. Choose **Spark SQL** for the node type.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, or if you want multiple inputs for the SQL transform, choose a node from the **Node parents** list to use as the input source for the transform. Add additional parent nodes as needed.
3. Choose the **Transform** tab in the node details panel.
4. The source datasets for the SQL query are identified by the names you specified in the **Name** field for each node. If you do not want to use these names, or if the names are not suitable for a SQL query, you can associate a name to each dataset. The console provides default aliases, such as `MyDataSource`.



For example, if a parent node for the SQL transform node is named `Rename Org PK field`, you might associate the name `org_table` with this dataset. This alias can then be used in the SQL query in place of the node name.

5. In the text entry field under the heading **Code block**, paste or enter the SQL query. The text field displays SQL syntax highlighting and keyword suggestions.
6. With the SQL transform node selected, choose the **Output schema** tab, and then choose **Edit**. Provide the columns and data types that describe the output fields of the SQL query.

Specify the schema using the following actions in the **Output schema** section of the page:

- To rename a column, place the cursor in the **Key** text box for the column (also referred to as a *field* or *property key*) and enter the new name.
 - To change the data type for a column, select the new data type for the column from the drop-down list.
 - To add a new top-level column to the schema, choose the Overflow () button, and then choose **Add root key**. New columns are added at the top of the schema.
 - To remove a column from the schema, choose the delete icon () to the far right of the Key name.
7. When you finish specifying the output schema, choose **Apply** to save your changes and exit the schema editor. If you do not want to save your changes, choose **Cancel** to edit the schema editor.
 8. (Optional) After configuring the node properties and transform properties, you can preview the modified dataset by choosing the **Data preview** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. There is a cost associated with using this feature, and billing starts as soon as you provide an IAM role.

Creating a custom transformation

If you need to perform more complicated transformations on your data, or want to add data property keys to the dataset, you can add a **Custom code** transform to your job diagram. The Custom code node allows you to enter a script that performs the transformation.

When using custom code, you must use a schema editor to indicate the changes made to the output through the custom code. When editing the schema, you can perform the following actions:

- Add or remove data property keys

- Change the data type of data property keys
- Change the name of data property keys
- Restructure a nested property key

You must use a *SelectFromCollection* transform to choose a single `DynamicFrame` from the result of your Custom transform node before you can send the output to a target location.

Use the following tasks to add a custom transform node to your job diagram.

Adding a custom code transform node to the job diagram

To add a custom transform node to your job diagram

1. (Optional) Choose **Transform** in the toolbar at the top of the visual editor, and then choose **Custom transform** to add a custom transform to your job diagram.
2. On the **Node properties** tab, enter a name for the node in the job diagram. If a node parent is not already selected, or if you want multiple inputs for the custom transform, then choose a node from the **Node parents** list to use as the input source for the transform.

Entering code for the custom transform node

You can type or copy code into an input field. The job uses this code to perform the data transformation. You can provide a code snippet in either Python or Scala. The code should take one or more `DynamicFrames` as input and returns a collection of `DynamicFrames`.

To enter the script for a custom transform node

1. With the custom transform node selected in the job diagram, choose the **Transform** tab.
2. In the text entry field under the heading **Code block**, paste or enter the code for the transformation. The code that you use must match the language specified for the job on the **Job details** tab.

When referring to the input nodes in your code, AWS Glue Studio names the `DynamicFrames` returned by the job diagram nodes sequentially based on the order of creation. For example:

- Data source nodes: `DataSource0`, `DataSource1`, `DataSource2`, and so on.
- Transform nodes: `Transform0`, `Transform1`, `Transform2`, and so on.

The following examples show the format of the code to enter in the code box:

Python

The following example takes the first `DynamicFrame` received, converts it to a `DataFrame` to apply the native filter method (keeping only records that have over 1000 votes), then converts it back to a `DynamicFrame` before returning it.

```
def FilterHighVoteCounts (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF()
    df_filtered = df.filter(df["vote_count"] > 1000)
    dyf_filtered = DynamicFrame.fromDF(df_filtered, glueContext, "filter_votes")
    return(DynamicFrameCollection({"CustomTransform0": dyf_filtered}, glueContext))
```

Scala

The following example takes the first `DynamicFrame` received, converts it to a `DataFrame` to apply the native filter method (keeping only records that have over 1000 votes), then converts it back to a `DynamicFrame` before returning it.

```
object FilterHighVoteCounts {  
  def execute(glueContext : GlueContext, input : Seq[DynamicFrame]) : Seq[DynamicFrame]  
  = {  
    val frame = input(0).toDF()  
    val filtered = DynamicFrame(frame.filter(frame("vote_count") > 1000), glueContext)  
    Seq(filtered)  
  }  
}
```



Editing the schema in a custom transform node






When you use a custom transform node, AWS Glue Studio cannot automatically infer the output schemas created by the transform. You use the schema editor to describe the schema changes implemented by the custom transform code.


A custom code node can have any number of parent nodes, each providing a `DynamicFrame` as input for your custom code. A custom code node returns a collection of `DynamicFrames`. Each `DynamicFrame` that is used as input has an associated schema. You must add a schema that describes each `DynamicFrame` returned by the custom code node.

To edit the output schemas for a custom transform node

1. With the custom transform node selected in the job diagram, in the node details panel, choose the **Output schema** tab.
2. Choose **Edit** to make changes to the schema.

If you have nested data property keys, such as an array or object, you can choose the **Expand-Rows** icon () on the top right of each schema panel to expand the list of child data property keys. After you choose this icon, it changes to the **Collapse-Rows** icon (), which you can choose to collapse the list of child property keys.

3. Modify the schema using the following actions in the section on the right side of the page:
 - To rename a property key, place the cursor in the **Key** text box for the property key, then enter the new name.
 - To change the data type for a property key, use the list to choose the new data type for the property key.
 - To add a new top-level property key to the schema, choose the **Overflow** () icon to the left of the **Cancel** button, and then choose **Add root key**.
 - To add a child property key to the schema, choose the **Add-Key** icon  associated with the parent key. Enter a name for the child key and choose the data type.
 - To remove a property key from the schema, choose the **Remove** icon () to the far right of the key name.
4. If your custom transform code uses multiple `DynamicFrames`, you can add additional output schemas.
 - To add a new, empty schema, choose the **Overflow** () icon, and then choose **Add output schema**.
 - To copy an existing schema to a new output schema, make sure the schema you want to copy is displayed in the schema selector. Choose the **Overflow** () icon, and then choose **Duplicate**.

If you want to remove an output schema, make sure the schema you want to copy is displayed in the schema selector. Choose the **Overflow** () icon, and then choose **Delete**.

5. Add new root keys to the new schema or edit the duplicated keys.
6. When you are modifying the output schemas, choose the **Apply** button to save your changes and exit the schema editor.

If you do not want to save your changes, choose the **Cancel** button.

Configure the custom transform output

A custom code transform returns a collection of `DynamicFrames`, even if there is only one `DynamicFrame` in the result set.

To process the output from a custom transform node

1. Add a `SelectFromCollection` transform node, which has the custom transform node as its parent node. Update this transform to indicate which dataset you want to use. See [Using `SelectFromCollection` to choose which dataset to keep](#) (p. 32) for more information.
2. Add additional `SelectFromCollection` transforms to the job diagram if you want to use additional `DynamicFrames` produced by the custom transform node.

Consider a scenario in which you add a custom transform node to split a flight dataset into multiple datasets, but duplicate some of the identifying property keys in each output schema, such as the flight date or flight number. You add a `SelectFromCollection` transform node for each output schema, with the custom transform node as its parent.

3. (Optional) You can then use each `SelectFromCollection` transform node as input for other nodes in the job, or as a parent for a data target node.

Configuring data target nodes

The data target is where the job writes the transformed data.

Overview of data target options

Your data target (also called a *data sink*) can be:

- **S3** – The job writes the data in a file in the Amazon S3 location you choose and in the format you specify.

If you configure partition columns for the data target, then the job writes the dataset to Amazon S3 into directories based on the partition key.

- **AWS Glue Data Catalog** – The job uses the information associated with the table in the Data Catalog to write the output data to a target location.

You can create the table manually or with the crawler. You can also use AWS CloudFormation templates to create tables in the Data Catalog.

- A connector – A connector is a piece of code that facilitates communication between your data store and AWS Glue. The job uses the connector and associated connection to write the output data to a target location. You can either subscribe to a connector offered in AWS Marketplace, or you can create your own custom connector. For more information, see [Adding connectors to AWS Glue Studio](#) (p. 47)

You can choose to update the Data Catalog when your job writes to an Amazon S3 data target. Instead of requiring a crawler to update the Data Catalog when the schema or partitions change, this option

makes it easy to keep your tables up to date. This option simplifies the process of making your data available for analytics by optionally adding new tables to the Data Catalog, updating table partitions, and updating the schema of your tables directly from the job.

Editing the data target node

The data target is where the job writes the transformed data.

To add or configure a data target node in your job diagram

1. (Optional) If you need to add a target node, choose **Target** in the toolbar at the top of the visual editor, and then choose either **S3** or **Glue Data Catalog**.
 - If you choose **S3** for the target, then the job writes the dataset to one or more files in the Amazon S3 location you specify.
 - If you choose **AWS Glue Data Catalog** for the target, then the job writes to a location described by the table selected from the Data Catalog.
2. Choose a data target node in the job diagram. When you choose a node, the node details panel appears on the right-side of the page.
3. Choose the **Node properties** tab, and then enter the following information:
 - **Name:** Enter a name to associate with the node in the job diagram.
 - **Node type:** A value should already be selected, but you can change it as needed.
 - **Node parents:** The parent node is the node in the job diagram that provides the output data you want to write to the target location. For a pre-populated job diagram, the target node should already have the parent node selected. If there is no parent node displayed, then choose a parent node from the list.

A target node has a single parent node.
4. Configure the **Data target properties** information. For more information, see the following sections:
 - [Using Amazon S3 for the data target \(p. 39\)](#)
 - [Using Data Catalog tables for the data target \(p. 40\)](#)
 - [Using a connector for the data target \(p. 41\)](#)
5. (Optional) After configuring the data target node properties, you can view the output schema for your data by choosing the **Output schema** tab in the node details panel. The first time you choose this tab for any node in your job, you are prompted to provide an IAM role to access the data. If you have not specified an IAM role on the **Job details** tab, you are prompted to enter an IAM role here.

Using Amazon S3 for the data target

For all data sources except Amazon S3 and connectors, a table must exist in the AWS Glue Data Catalog for the source type that you choose. AWS Glue Studio does not create the Data Catalog table.

To configure a data target node that writes to Amazon S3

1. Go to the visual editor for a new or saved job.
2. Choose a data source node in the job diagram.
3. Choose the **Data source properties** tab, and then enter the following information:
 - **Format:** Choose a format from the list. The available format types for the data results are:
 - **JSON:** JavaScript Object Notation.
 - **CSV:** Comma-separated values.

- **Avro:** Apache Avro JSON binary.
- **Parquet:** Apache Parquet columnar storage.
- **Glue Parquet:** A custom Parquet writer type that is optimized for `DynamicFrames` as the data format. Instead of requiring a precomputed schema for the data, it computes and modifies the schema dynamically.
- **ORC:** Apache Optimized Row Columnar (ORC) format.

To learn more about these format options, see [Format Options for ETL Inputs and Outputs in AWS Glue](#) in the *AWS Glue Developer Guide*.

- **Compression Type:** You can choose to optionally compress the data using either the `gzip` or `bzip2` format. The default is no compression, or **None**.
- **S3 Target Location:** The Amazon S3 bucket and location for the data output. You can choose the **Browse S3** button to see the Amazon S3 buckets that you have access to and choose one as the target destination.
- **Data catalog update options**
 - **Do not update the Data Catalog:** (Default) Choose this option if you don't want the job to update the Data Catalog, even if the schema changes or new partitions are added.
 - **Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions:** If you choose this option, the job creates the table in the Data Catalog on the first run of the job. On subsequent job runs, the job updates the Data Catalog table if the schema changes or new partitions are added.

You must also select a database from the Data Catalog and enter a table name.

- **Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions:** If you choose this option, the job creates the table in the Data Catalog on the first run of the job. On subsequent job runs, the job updates the Data Catalog table only to add new partitions.

You must also select a database from the Data Catalog and enter a table name.

- **Partition keys:** Choose which columns to use as partitioning keys in the output. To add more partition keys, choose **Add a partition key**.

Using Data Catalog tables for the data target

For all data sources except Amazon S3 and connectors, a table must exist in the AWS Glue Data Catalog for the target type that you choose. AWS Glue Studio does not create the Data Catalog table.

To configure the data properties for a target that uses a Data Catalog table

1. Go to the visual editor for a new or saved job.
2. Choose a data target node in the job diagram.
3. Choose the **Data target properties** tab, and then enter the following information:
 - **Database:** Choose the database that contains the table you want to use as the target from the list. This database must already exist in the Data Catalog.
 - **Table:** Choose the table that defines the schema of your output data from the list. This table must already exist in the Data Catalog.

A table in the Data Catalog consists of the names of columns, data type definitions, partition information, and other metadata about the target dataset. Your job writes to a location described by this table in the Data Catalog.

For more information about creating tables in the Data Catalog, see [Defining Tables in the Data Catalog](#) in the *AWS Glue Developer Guide*.

- **Data catalog update options**

- **Do not change table definition:** (Default) Choose this option if you don't want the job to update the Data Catalog, even if the schema changes, or new partitions are added.
- **Update schema and add new partitions:** If you choose this option, the job updates the Data Catalog table if the schema changes or new partitions are added.
- **Keep existing schema and add new partitions:** If you choose this option, the job updates the Data Catalog table only to add new partitions.
- **Partition keys:** Choose which columns to use as partitioning keys in the output. To add more partition keys, choose **Add a partition key**.

Using a connector for the data target

If you select a connector for the **Node type**, follow the instructions at [Authoring jobs with custom connectors \(p. 50\)](#) to finish configuring the data target properties.

Editing or uploading a job script

Use the AWS Glue Studio visual editor to edit the job script or upload your own script.

You can use the visual editor to edit job nodes only if the jobs were created with AWS Glue Studio. If the job was created using the AWS Glue console, through API commands, or with the command line interface (CLI), you can use the script editor in AWS Glue Studio to edit the job script, parameters, and schedule. You can also edit the script for a job created in AWS Glue Studio by converting the job to script-only mode.

To edit the job script or upload your own script

1. If creating a new job, on the **Jobs** page, choose the **Spark script editor** option to create a Spark job or choose the **Python Shell script editor** to create a Python shell job. You can either write a new script, or upload an existing script. If you choose **Spark script editor**, you can write or upload either a Scala or Python script. If you choose **Python Shell script editor**, you can only write or upload a Python script.

After choosing the option to create a new job, in the **Options** section that appears, you can choose to either start with a starter script (**Create a new script with boilerplate code**), or you can upload a local file to use as the job script.

If you chose **Spark script editor**, you can upload either Python or Scala script files. Scala scripts must have the file extension `.scala`. Python scripts must be recognized as files of type Python. If you chose **Python Shell script editor**, you can upload only Python script files.

When you are finished making your choices, choose **Create** to create the job and open the visual editor.

2. Go to the visual job editor for the new or saved job, and then choose the **Script** tab.
3. If you didn't create a new job using one of the script editor options, and you have never edited the script for an existing job, the **Script** tab displays the heading **Script (Locked)**. This means the script editor is in read-only mode. Choose **Edit script** to unlock the script for editing.

To make the script editable, AWS Glue Studio converts your job from a visual job to a script-only job. If you unlock the script for editing, you can't use the visual editor anymore for this job after you save it.

In the confirmation window, choose **Confirm** to continue or **Cancel** to keep the job available for visual editing.

If you choose **Confirm**, the **Visual** tab no longer appears in the editor. You can use AWS Glue Studio to modify the script using the script editor, modify the job details or schedule, or view job runs.

Note

Until you save the job, the conversion to a script-only job is not permanent. If you refresh the console web page, or close the job before saving it and reopen it in the visual editor, you will still be able to edit the individual nodes in the visual editor.

4. Edit the script as needed.

When you are done editing the script, choose **Save** to save the job and permanently convert the job from visual to script-only.

5. (Optional) You can download the script from the AWS Glue Studio console by choosing the **Download** button on the **Script** tab. When you choose this button, a new browser window opens, displaying the script from its location in Amazon S3. The **Script filename** and **Script path** parameters in the **Job details** tab of the job determine the name and location of the script file in Amazon S3.

Join test job2

The screenshot shows the 'Job details' tab in AWS Glue Studio. At the top, there are five tabs: 'Visual', 'Script', 'Job details' (which is selected and highlighted in orange), 'Runs', and 'Schedules'. Below the tabs, there is a section titled 'Advanced properties' with a downward arrow. Under this section, there are two main fields: 'Script filename' and 'Script path'. The 'Script filename' field contains the text 'Join test job.py'. The 'Script path' field has a description: 'S3 location of the script. Path must be in the form s3://bucket/prefix/path/. It must end with a slash (/) and not include any files.' Below the description, there is a search bar containing 's3://aws-glue-assets-111122223333-t' with a magnifying glass icon on the left and a close 'X' icon on the right. To the right of the search bar are two buttons: 'View' with an external link icon and 'Browse S3'. Below these fields, there are three checkboxes with labels and 'Info' links: 'Job metrics' (unchecked), 'Continuous logging' (checked), and 'Spark UI' (checked). Each checkbox has a description below it: 'Enable the creation of CloudWatch metrics when this job runs.', 'Enable logs in CloudWatch.', and 'Enable using Spark UI for monitoring this job.' respectively.

When you save the job, AWS Glue save the job script at the location specified by these fields. If you modify the script file at this location within Amazon S3, AWS Glue Studio will load the modified script the next time you edit the job.

Creating and editing Scala scripts in AWS Glue Studio

When you choose the script editor for creating a job, by default, the job programming language is set to `Python 3`. If you choose to write a new script instead of uploading a script, AWS Glue Studio starts a new script with boilerplate text written in Python. If you want to write a Scala script instead, you must first configure the script editor to use Scala.

Note

If you choose Scala as the programming language for the job and use the visual editor to design your job, the generated job script is written in Scala, and no further actions are needed.

To write a new Scala script in AWS Glue Studio

1. Create a new job by choosing the **Spark script editor** option.
2. Under **Options**, choose **Create a new script with boilerplate code**.
3. Choose the **Job details** tab and set **Language** to Scala (instead of Python 3).

Note

The **Type** property for the job is automatically set to Spark when you choose the **Spark script editor** option to create a job.

4. Choose the **Script** tab.
5. Remove the Python boilerplate text. You can replace it with the following Scala boilerplate text.

```
import com.amazonaws.services.glue.{DynamicRecord, GlueContext}
import org.apache.spark.SparkContext
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job

object MyScript {
  def main(args: Array[String]): Unit = {
    val sc: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(sc)

  }
}
```

6. Write your Scala job script in the editor. Add additional `import` statements as needed.

Creating and editing Python shell jobs in AWS Glue Studio

When you choose the Python shell script editor for creating a job, you can upload an existing Python script, or write a new one. If you choose to write a new script, boilerplate code is added to the new Python job script.

To create a new Python shell job

Refer to the instructions at [Start the job creation process \(p. 16\)](#).

The job properties that are supported for Python shell jobs are not the same as those supported for Spark jobs. The following list describes the changes to the available job parameters for Python shell jobs on the **Job details** tab.

- The **Type** property for the job is automatically set to `Python Shell` and can't be changed.
- Instead of **Language**, there is a **Python version** property for the job. Currently, Python shell jobs created in AWS Glue Studio use Python 3.6.
- The **Glue version** property is not available, because it does not apply to Python shell jobs.
- Instead of **Worker type** and **Number of workers**, a **Data processing units** property is shown instead. This job property determines how many data processing units (DPUs) are consumed by the Python shell when running the job.
- The **Job bookmark** property is not available, because it is not supported for Python shell jobs.
- Under **Advanced properties**, the following properties are not available for Python shell jobs.
 - **Job metrics**
 - **Continuous logging**

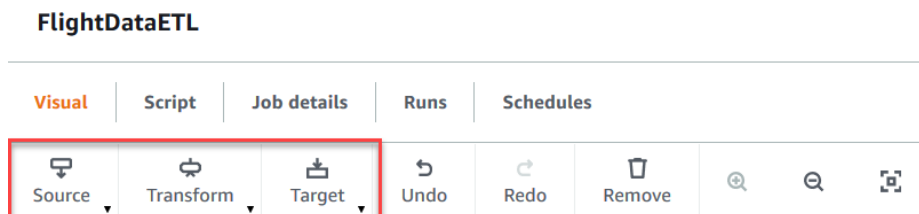
- **Spark UI** and **Spark UI logs path**
- **Dependent jars path**, under the heading **Libraries**

Adding nodes to the job diagram

You can add additional data sources, transforms, and data targets to your job to support more complex ETL actions.

To add nodes to a job diagram

1. Go to the visual editor for a new or saved job and choose the **Visual** tab.
2. Use the toolbar buttons to add a node of a specific type: **Source**, **Transform**, or **Target**.



3. Edit the node, as described in the following sections:
 - For a source node, see [Editing the data source node \(p. 20\)](#).
 - For a transform node, see [Editing the data transform node \(p. 25\)](#).
 - For a data target node, see [Editing the data target node \(p. 39\)](#).
4. If you're inserting a node in between two nodes in the job diagram, then perform the following actions:
 - a. Choose the node that will be the parent for the new node.
 - b. Choose one of the toolbar buttons to add a new node to the job diagram. The new node is added as a child of the currently selected node.
 - c. Choose the node that will be the child of the newly added node and change its parent node to point to the newly added node.

If you added a node by mistake, you can use the **Undo** button on the toolbar to reverse the action.

Changing the parent nodes for a node in the job diagram

You can change a node's parents to move nodes within the job diagram or to change a data source for a node.

To change the parent node

1. Choose the node in the job diagram that you want to modify.
2. In the node details panel, on the **Node properties** tab, under the heading **Node parents** remove the current parent for the node.
3. Choose a new parent node from the list.
4. Modify the other properties of the node as needed to match the newly selected parent node.

If you modified a node by mistake, you can use the **Undo** button on the toolbar to reverse the action.

Deleting nodes from the job diagram

You can remove nodes from the job diagram.

To remove a node

1. Go to the visual editor for a new or saved job and choose the **Visual** tab.
2. Choose the node you want to remove.
3. In the toolbar at the top of the visual editing pane, choose the **Remove** button.
4. If the node you removed had children nodes, modify the parents for those nodes as needed.

If you removed a node by mistake, you can use the **Undo** button on the toolbar to reverse the action.

Using connectors and connections with AWS Glue Studio

AWS Glue provides built-in support for the most commonly used data stores (such as Amazon Redshift, Amazon Aurora, Microsoft SQL Server, MySQL, MongoDB, and PostgreSQL) using JDBC connections. AWS Glue also allows you to use custom JDBC drivers in your extract, transform, and load (ETL) jobs. For data stores that are not natively supported, such as SaaS applications, you can use connectors.

A *connector* is an optional code package that assists with accessing data stores in AWS Glue Studio. You can subscribe to several connectors offered in AWS Marketplace.

When creating ETL jobs, you can use a natively supported data store, a connector from AWS Marketplace, or your own custom connectors. If you use a connector, you must first create a connection for the connector. A *connection* contains the properties that are required to connect to a particular data store. You use the connection with your data sources and data targets in the ETL job. Connectors and connections work together to facilitate access to the data stores.

Topics

- [Overview of using connectors and connections \(p. 46\)](#)
- [Adding connectors to AWS Glue Studio \(p. 47\)](#)
- [Creating connections for connectors \(p. 50\)](#)
- [Authoring jobs with custom connectors \(p. 50\)](#)
- [Managing connectors and connections \(p. 55\)](#)
- [Developing custom connectors \(p. 57\)](#)
- [Restrictions for using connectors and connections in AWS Glue Studio \(p. 59\)](#)

Overview of using connectors and connections

A *connection* contains the properties that are required to connect to a particular data store. When you create a connection, it is stored in the AWS Glue Data Catalog. You choose a connector, and then create a connection based on that connector.

You can subscribe to connectors for non-natively supported data stores in AWS Marketplace, and then use those connectors when you're creating connections. Developers can also create their own connectors, and you can use them when creating connections.

Note

Connections created using the AWS Glue console do not appear in AWS Glue Studio. Connections created using custom or AWS Marketplace connectors in AWS Glue Studio appear in the AWS Glue console with type set to UNKNOWN.

The following steps describe the overall process of using connectors in AWS Glue Studio:

1. Subscribe to a connector in AWS Marketplace, or develop your own connector and upload it to AWS Glue Studio. For more information, see [Adding connectors to AWS Glue Studio \(p. 47\)](#).
2. Review the connector usage information. You can find this information on the **Usage** tab on the connector product page. For example, if you click the **Usage** tab on this product page, [AWS Glue](#)

[Connector for Google BigQuery](#), you can see in the **Additional Resources** section a link to a blog about using this connector. Other connectors might contain links to the instructions in the **Overview** section, as shown on the connector product page for [Cloudwatch Logs connector for AWS Glue](#).

3. Create a connection. You choose which connector to use and provide additional information for the connection, such as login credentials, URI strings, and virtual private cloud (VPC) information. For more information, see [Creating connections for connectors](#) (p. 50).
4. Create an IAM role for your job. The job assumes the permissions of the IAM role that you specify when you create it. This IAM role must have the necessary permissions to authenticate with, extract data from, and write data to your data stores. For more information, see [Overview of Job-related permissions](#) (p. 8) and [Permissions required for using connectors](#) (p. 9).
5. Create an ETL job and configure the data source properties for your ETL job. Provide the connection options and authentication information as instructed by the custom connector provider. For more information, see [Authoring jobs with custom connectors](#) (p. 50).
6. Customize your ETL job by adding transforms or additional data stores, as described in [Editing ETL jobs in AWS Glue Studio](#) (p. 18).
7. If using a connector for the data target, configure the data target properties for your ETL job. Provide the connection options and authentication information as instructed by the custom connector provider. For more information, see [the section called "Authoring jobs with custom connectors"](#) (p. 50).
8. Customize the job run environment by configuring job properties, as described in [Modify the job properties](#) (p. 69).
9. Run the job.

Adding connectors to AWS Glue Studio

A connector is a piece of code that facilitates communication between your data store and AWS Glue. You can either subscribe to a connector offered in AWS Marketplace, or you can create your own custom connector.

Topics

- [Subscribing to AWS Marketplace connectors](#) (p. 47)
- [Creating custom connectors](#) (p. 48)

Subscribing to AWS Marketplace connectors

AWS Glue Studio makes it easy to add connectors from AWS Marketplace.

To add a connector from AWS Marketplace to AWS Glue Studio

1. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
2. On the **Connectors** page, choose **Go to AWS Marketplace**.
3. In AWS Marketplace, in **Featured products**, choose the connector you want to use. You can choose one of the featured connectors, or use search. You can search on the name or type of connector, and you can use options to refine the search results.

If you want to use one of the featured connectors, choose **View product**. If you used search to locate a connector, then choose the name of the connector.

4. On the product page for the connector, use the tabs to view information about the connector. If you decide to purchase this connector, choose **Continue to Subscribe**.
5. Provide the payment information, and then choose **Continue to Configure**.

6. On the **Configure this software** page, choose the method of deployment and the version of the connector to use. Then choose **Continue to Launch**.
7. On the **Launch this software** page, you can review the **Usage Instructions** provided by the connector provider. When you're ready to continue, choose **Activate connection in AWS Glue Studio**.

After a small amount of time, the console displays the **Create marketplace connection** page in AWS Glue Studio.

8. Create a connection that uses this connector, as described in [Creating connections for connectors \(p. 50\)](#).

Alternatively, you can choose **Activate connector only** to skip creating a connection at this time. You must create a connection at a later date before you can use the connector.

Creating custom connectors

You can also build your own connector and then upload the connector code to AWS Glue Studio.

Custom connectors are integrated into AWS Glue Studio through the AWS Glue Spark runtime API. The AWS Glue Spark runtime allows you to plug in any connector that is compliant with the Spark, Athena, or JDBC interface. It allows you to pass in any connection option that is available with the custom connector.

You can encapsulate all your connection properties with [AWS Glue Connections](#) and supply the connection name to your ETL job. Integration with Data Catalog connections allows you to use the same connection properties across multiple calls in a single Spark application or across different applications.

You can specify additional options for the connection. The job script that AWS Glue Studio generates contains a `Datasource` entry that uses the connection to plug in your connector with the specified connection options. For example:

```
Datasource = glueContext.create_dynamic_frame.from_options(connection_type =  
"custom.jdbc", connection_options = {"dbTable":"Account", "connectionName":"my-custom-jdbc-  
connection"}, transformation_ctx = "DataSource0")
```

To add a custom connector to AWS Glue Studio

1. Create the code for your custom connector. For more information, see [Developing custom connectors \(p. 57\)](#).
2. Add support for AWS Glue features to your connector. Here are some examples of these features and how they are used within the job script generated by AWS Glue Studio:

- **Data type mapping** – Your connector can typecast the columns while reading them from the underlying data store. For example, a `dataTypeMapping` of `{"INTEGER": "STRING"}` converts all columns of type `Integer` to columns of type `String` when parsing the records and constructing the `DynamicFrame`. This helps users to cast columns to types of their choice.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type  
= "custom.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"}",  
connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- **Partitioning for parallel reads** – AWS Glue allows parallel data reads from the data store by partitioning the data on a column. You must specify the partition column, the lower partition bound, the upper partition bound, and the number of partitions. This feature enables you to make use of data parallelism and multiple Spark executors allocated for the Spark application.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type = "custom.jdbc", connection_options = {"upperBound":"200","numPartitions":"4", "partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- **Use AWS Secrets Manager for storing credentials** – The Data Catalog connection can also contain a `secretId` for a secret stored in AWS Secrets Manager. The AWS secret can securely store authentication and credentials information and provide it to your ETL job at runtime. Alternatively, you can specify the `secretId` from the Spark script as follows:

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type = "custom.jdbc", connection_options = {"connectionName":"test-connection-jdbc", "secretId"-> "my-secret-id"}, transformation_ctx = "DataSource0")
```

- **Filtering the source data with row predicates and column projections** – The AWS Glue Spark runtime also allows users to push down SQL queries to filter data at the source with row predicates and column projections. This allows your ETL job to load filtered data faster from data stores that support push-downs. An example SQL query pushed down to a JDBC data source is:
`SELECT id, name, department FROM department WHERE id < 200.`

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type = "custom.jdbc", connection_options = {"query":"SELECT id, name, department FROM department WHERE id < 200","connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- **Job bookmarks** – AWS Glue supports incremental loading of data from JDBC sources. AWS Glue keeps track of the last processed record from the data store, and processes new data records in the subsequent ETL job runs. Job bookmarks use the primary key as the default column for the bookmark key, provided that this column increases or decreases sequentially. For more information about job bookmarks, see [Job Bookmarks](#) in the *AWS Glue Developer Guide*.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type = "custom.jdbc", connection_options = {"jobBookmarkKeys":["empno"], "jobBookmarkKeysSortOrder":"asc", "connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

3. Package the custom connector as a JAR file and upload the file to Amazon S3.
4. Test your custom connector. For more information, see the instructions on GitHub at [Blue Custom Connectors: Local Validation Tests Guide](#).
5. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
6. On the **Connectors** page, choose **Create custom connector**.
7. On the **Create custom connector** page, enter the following information:
 - The path to the location of the custom code JAR file in Amazon S3.
 - A name for the connector that will be used by AWS Glue Studio.
 - Your connector type, which can be one of **JDBC**, **Spark**, or **Athena**.
 - The name of the entry point within your custom code that AWS Glue Studio calls to use the connector.
 - For JDBC connectors, this field should be the class name of your JDBC driver.
 - For Spark connectors, this field should be the fully qualified data source class name, or its alias, that you use when loading the Spark data source with the `format` operator.
 - (JDBC only) The base URL used by the JDBC connection for the data store.
 - (Optional) A description of the custom connector.
8. Choose **Create connector**.

9. From the **Connectors** page, create a connection that uses this connector, as described in [Creating connections for connectors](#) (p. 50).

Creating connections for connectors

An AWS Glue connection is a Data Catalog object that stores connection information for a particular data store. Connections store login credentials, URI strings, virtual private cloud (VPC) information, and more. Creating connections in the Data Catalog saves the effort of having to specify all connection details every time you create a job.

Note

Connections created using the AWS Glue console do not appear in AWS Glue Studio.

To create a connection for a connector

1. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
2. Choose the connector you want to create a connection for, and then choose **Create connection**.
3. On the **Create connection** page, enter a name for your connection, and optionally a description.
4. Enter the connection details. Depending on the type of connector you selected, you're prompted to enter additional information:
 - Enter the requested authentication information, such as a user name and password, or choose an AWS secret.
 - For connectors that use JDBC, enter the information required to create the JDBC URL for the data store.
 - If you use a virtual private cloud (VPC), then enter the network information for your VPC.
5. Choose **Create connection**.

You are returned to the **Connectors** page, and the informational banner indicates the connection that was created. You can now use the connection in your AWS Glue Studio jobs, as described in [the section called "Create jobs that use a connector" \(p. 17\)](#).

Authoring jobs with custom connectors

You can use connectors and connections for both data source nodes and data target nodes in AWS Glue Studio.

Topics

- [Create jobs that use a connector for the data source](#) (p. 50)
- [Configure source properties for nodes that use connectors](#) (p. 51)
- [Configure target properties for nodes that use connectors](#) (p. 54)

Create jobs that use a connector for the data source

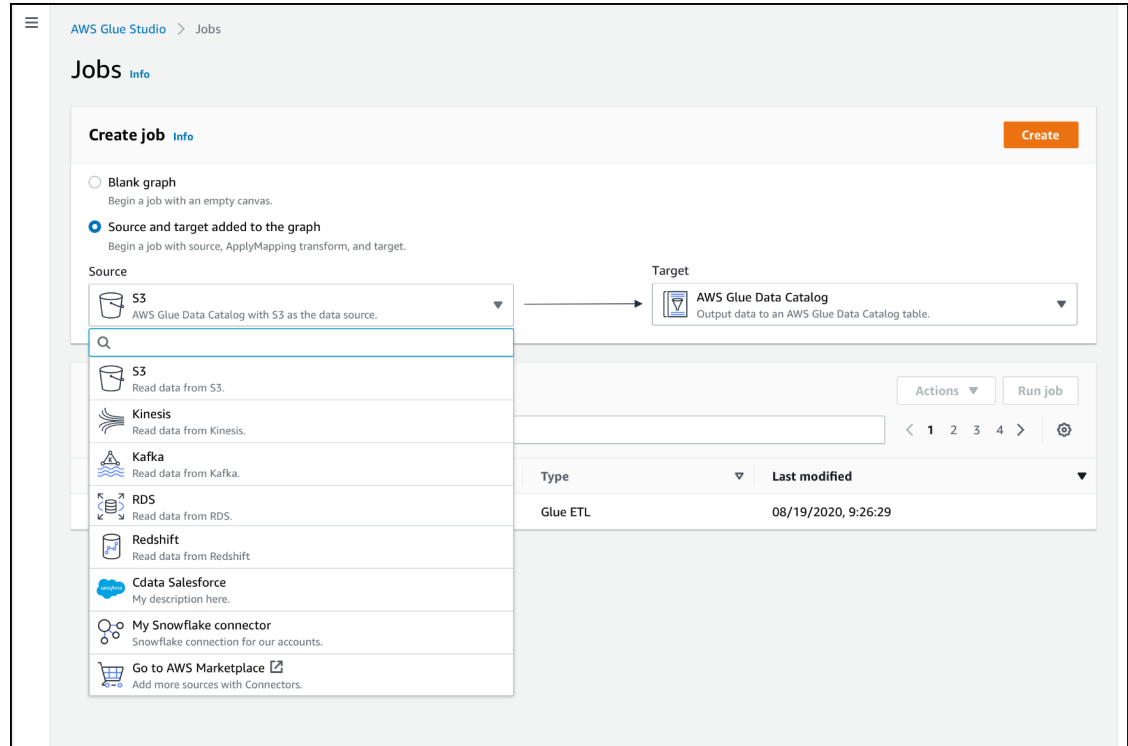
When you create a new job, you can choose a connector for the data source and data targets.

To create a job that uses connectors for the data source or data target

1. Sign in to the AWS Management Console and open the AWS Glue Studio console at <https://console.aws.amazon.com/gluestudio/>.

2. On the **Connectors** page, in the **Your connections** resource list, choose the connection you want to use in your job, and then choose **Create job**.

Alternatively, on the AWS Glue Studio **Jobs** page, under **Create job**, choose **Source and target added to the graph**. In the **Source** drop-down list, choose the custom connector that you want to use in your job. You can also choose a connector for **Target**.



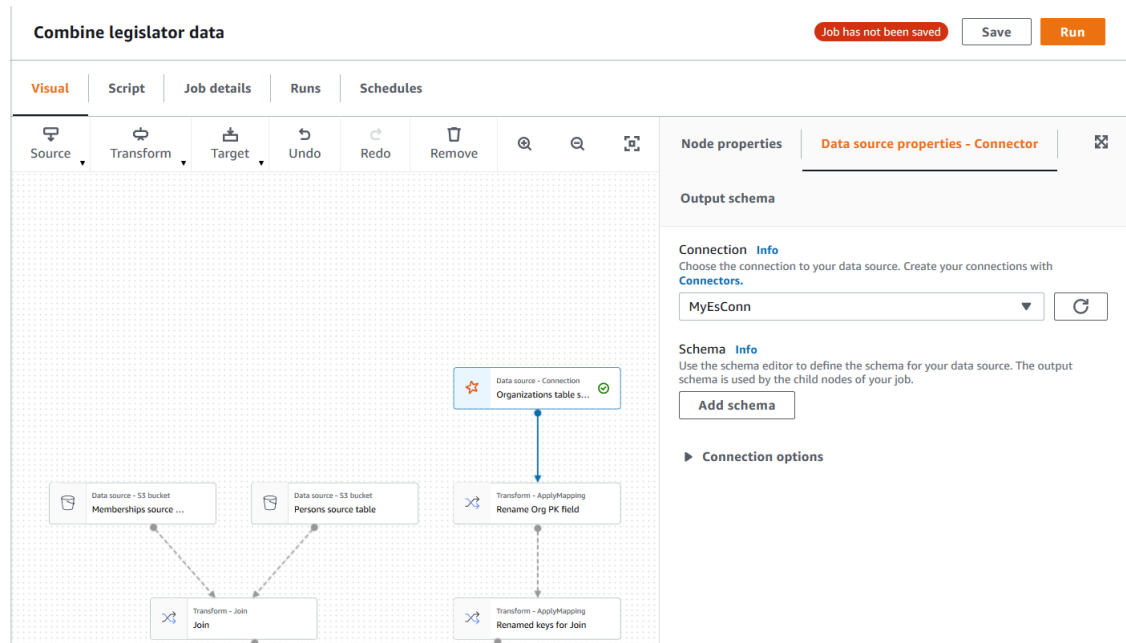
3. Choose **Create** to open the visual job editor.
4. Configure the data source node, as described in [Configure source properties for nodes that use connectors \(p. 51\)](#).
5. Continue creating your ETL job by adding transforms, additional data stores, and data targets, as described in [Editing ETL jobs in AWS Glue Studio \(p. 18\)](#).
6. Customize the job run environment by configuring job properties as described in [Modify the job properties \(p. 69\)](#).
7. Save and run the job.

Configure source properties for nodes that use connectors

After you create a job that uses a connector for the data source, the visual job editor displays a job graph with a data source node configured for the connector. You must configure the data source properties for that node.

To configure the properties for a data source node that uses a connector

1. Choose the connector data source node in the job graph or add a new node and choose the connector for the **Node type**. Then, on the right-side, in the node details panel, choose the **Data source properties** tab, if it's not already selected.



2. In the **Data source properties** tab, choose the connection that you want to use for this job.

Enter the additional information required for each connection type:

JDBC

- **Data source input type:** Choose to provide either a table name or a SQL query as the data source. Depending on your choice, you then need to provide the following additional information:
 - **Table name:** The name of the table in the data source. If the data source does not use the term *table*, then supply the name of an appropriate data structure, as indicated by the custom connector usage information (which is available in AWS Marketplace).
 - **Filter predicate:** A condition clause to use when reading the data source, similar to a *WHERE* clause, which is used to retrieve a subset of the data.
 - **Query code:** Enter a SQL query to use to retrieve a specific dataset from the data source. An example of a basic SQL query is:

```
SELECT column_list FROM table_name WHERE where_clause
```

- **Schema:** Because AWS Glue Studio is using information stored in the connection to access the data source instead of retrieving metadata information from a Data Catalog table, you must provide the schema metadata for the data source. Choose **Add schema** to open the schema editor.

For instructions on how to use the schema editor, see [Editing the schema in a custom transform node \(p. 37\)](#).

- **Partition column:** (Optional) You can choose to partition the data reads by providing values for **Partition column**, **Lower bound**, **Upper bound**, and **Number of partitions**.

The `lowerBound` and `upperBound` values are used to decide the partition stride, not for filtering the rows in table. All rows in the table are partitioned and returned.

Note

Column partitioning adds an extra partitioning condition to the query used to read the data. When using a query instead of a table name, you should validate that the query works with the specified partitioning condition. For example:

- If your query format is "SELECT col1 FROM table1", then test the query by appending a WHERE clause at the end of the query that uses the partition column.
- If your query format is "SELECT col1 FROM table1 WHERE col2=val", then test the query by extending the WHERE clause with AND and an expression that uses the partition column.
- **Data type casting:** If the data source uses data types that are not available in JDBC, use this section to specify how a data type from the data source should be converted into JDBC data types. You can specify up to 50 different data type conversions. All columns in the data source that use the same data type are converted in the same way.

For example, if you have three columns in the data source that use the `Float` data type, and you indicate that the `Float` data type should be converted to the JDBC `String` data type, then all three columns that use the `Float` data type are converted to `String` data types.

- **Job bookmark keys:** Job bookmarks help AWS Glue maintain state information and prevent the reprocessing of old data. Specify one more one or more columns as bookmark keys. AWS Glue Studio uses bookmark keys to track data that has already been processed during a previous run of the ETL job. Any columns you use for custom bookmark keys must be strictly monotonically increasing or decreasing, but gaps are permitted.

If you enter multiple bookmark keys, they're combined to form a single compound key. A compound job bookmark key should not contain duplicate columns. If you don't specify bookmark keys, AWS Glue Studio by default uses the primary key as the bookmark key, provided that the primary key is sequentially increasing or decreasing (with no gaps). If the table doesn't have a primary key, but the job bookmark property is enabled, you must provide custom job bookmark keys. Otherwise, the search for primary keys to use as the default will fail and the job run will fail.

- **Job bookmark keys sorting order:** Choose whether the key values are sequentially increasing or decreasing.

Spark

- **Schema:** Because AWS Glue Studio is using information stored in the connection to access the data source instead of retrieving metadata information from a Data Catalog table, you must provide the schema metadata for the data source. Choose **Add schema** to open the schema editor.

For instructions on how to use the schema editor, see [Editing the schema in a custom transform node \(p. 37\)](#).

- **Connection options:** Enter additional key-value pairs as needed to provide additional connection information or options. For example, you might enter a database name, table name, a user name, and password.

For example, for OpenSearch, you enter the following key-value pairs, as described in [Tutorial: Using the open-source Elasticsearch Spark Connector \(p. 60\)](#):

- `es.net.http.auth.user:username`
- `es.net.http.auth.pass:password`
- `es.nodes:https://<Elasticsearch endpoint>`
- `es.port:443`
- `path:<Elasticsearch resource>`
- `es.nodes.wan.only:true`

For an example of the minimum connection options to use, see the sample test script [MinimalSparkConnectorTest.scala](#) on GitHub, which shows the connection options you would normally provide in a connection.

Athena

- **Table name:** The name of the table in the data source. If you're using a connector for reading from Athena-CloudWatch logs, you would enter the table name `all_log_streams`.
- **Athena schema name:** Choose the schema in your Athena data source that corresponds to the database that contains the table. If you're using a connector for reading from Athena-CloudWatch logs, you would enter a schema name similar to `/aws/glue/name`.
- **Schema:** Because AWS Glue Studio is using information stored in the connection to access the data source instead of retrieving metadata information from a Data Catalog table, you must provide the schema metadata for the data source. Choose **Add schema** to open the schema editor.

For instructions on how to use the schema editor, see [Editing the schema in a custom transform node \(p. 37\)](#).

- **Additional connection options:** Enter additional key-value pairs as needed to provide additional connection information or options.

For an example, see the `README.md` file at <https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Athena>. In the steps in this document, the sample code shows the minimal required connection options, which are `tableName`, `schemaName`, and `className`. The code example specifies these options as part of the `optionsMap` variable, but you can specify them for your connection and then use the connection.

3. (Optional) After providing the required information, you can view the resulting data schema for your data source by choosing the **Output schema** tab in the node details panel. The schema displayed on this tab is used by any child nodes that you add to the job graph.

Configure target properties for nodes that use connectors

If you use a connector for the data target type, you must configure the properties of the data target node.

To configure the properties for a data target node that uses a connector

1. Choose the connector data target node in the job graph. Then, on the right-side, in the node details panel, choose the **Data target properties** tab, if it's not already selected.
2. In the **Data target properties** tab, choose the connection to use for writing to the target.

Enter the additional information required for each connection type:

JDBC

- **Connection:** Choose the connection to use with your connector. For information about how to create a connection, see [Creating connections for connectors \(p. 50\)](#).
- **Table name:** The name of the table in the data target. If the data target does not use the term *table*, then supply the name of an appropriate data structure, as indicated by the custom connector usage information (which is available in AWS Marketplace).

- **Batch size** (Optional): Enter the number of rows or records to insert in the target table in a single operation. The default value is 1000 rows.

Spark

- **Connection:** Choose the connection to use with your connector. If you did not create a connection previously, choose **Create connection** to create one. For information about how to create a connection, see [Creating connections for connectors](#) (p. 50).
- **Connection options:** Enter additional key-value pairs as needed to provide additional connection information or options. You might enter a database name, table name, a user name, and password.

For example, for OpenSearch, you enter the following key-value pairs, as described in [Tutorial: Using the open-source Elasticsearch Spark Connector](#) (p. 60):

- `es.net.http.auth.user`: `username`
- `es.net.http.auth.pass`: `password`
- `es.nodes`: `https://<Elasticsearch endpoint>`
- `es.port`: `443`
- `path`: `<Elasticsearch resource>`
- `es.nodes.wan.only`: `true`

For an example of the minimum connection options to use, see the sample test script [MinimalSparkConnectorTest.scala](#) on GitHub, which shows the connection options you would normally provide in a connection.

3. After providing the required information, you can view the resulting data schema for your data source by choosing the **Output schema** tab in the node details panel.

Managing connectors and connections

You use the **Connectors** page in AWS Glue Studio to manage your connectors and connections.

Topics

- [Viewing connector and connection details](#) (p. 55)
- [Editing connectors and connections](#) (p. 56)
- [Deleting connectors and connections](#) (p. 56)
- [Cancel a subscription for a connector](#) (p. 56)

Viewing connector and connection details

You can view summary information about your connectors and connections in the **Your connectors** and **Your connections** resource tables on the **Connectors** page. To view detailed information, perform the following steps.

Note

Connections created using the AWS Glue console do not appear in AWS Glue Studio.

To view connector or connection details

1. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
2. Choose the connector or connection that you want to view detailed information for.

3. Choose **Actions**, and then choose **View details** to open the detail page for that connector or connection.
4. On the detail page, you can choose to **Edit** or **Delete** the connector or connection.
 - For connectors, you can choose **Create connection** to create a new connection that uses the connector.
 - For connections, you can choose **Create job** to create a job that uses the connection.

Editing connectors and connections

You use the **Connectors** page to change the information stored in your connectors and connections.

To modify a connector or connection

1. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
2. Choose the connector or connection that you want to change.
3. Choose **Actions**, and then choose **Edit**.

You can also choose **View details** and on the connector or connection detail page, you can choose **Edit**.

4. On the **Edit connector** or **Edit connection** page, update the information, and then choose **Save**.

Deleting connectors and connections

You use the **Connectors** page to delete connectors and connections. If you delete a connector, then any connections that were created for that connector should also be deleted.

To remove connectors from AWS Glue Studio

1. In the AWS Glue Studio console, choose **Connectors** in the console navigation pane.
2. Choose the connector or connection you want to delete.
3. Choose **Actions**, and then choose **Delete**.

You can also choose **View details**, and on the connector or connection detail page, you can choose **Delete**.

4. Verify that you want to remove the connector or connection by entering **Delete**, and then choose **Delete**.

When deleting a connector, any connections that were created for that connector are also deleted.

Any jobs that use a deleted connection will no longer work. You can either edit the jobs to use a different data store, or remove the jobs. For information about how to delete a job, see [Delete jobs \(p. 73\)](#).

If you delete a connector, this doesn't cancel the subscription for the connector in AWS Marketplace. To remove a subscription for a deleted connector, follow the instructions in [Cancel a subscription for a connector \(p. 56\)](#).

Cancel a subscription for a connector

After you delete the connections and connector from AWS Glue Studio, you can cancel your subscription in AWS Marketplace if you no longer need the connector.

Note

If you cancel your subscription to a connector, this does not remove the connector or connection from your account. Any jobs that use the connector and related connections will no longer be able to use the connector and will fail.

Before you unsubscribe or re-subscribe to a connector from AWS Marketplace, you should delete existing connections and connectors associated with that AWS Marketplace product.

To unsubscribe from a connector in AWS Marketplace

1. Sign in to the AWS Marketplace console at <https://console.aws.amazon.com/marketplace>.
2. Choose **Manage subscriptions**.
3. On the **Manage subscriptions** page, choose **Manage** next to the connector subscription that you want to cancel.
4. Choose **Actions** and then choose **Cancel subscription**.
5. Select the check box to acknowledge that running instances are charged to your account, and then choose **Yes, cancel subscription**.

Developing custom connectors

You can write the code that reads data from or writes data to your data store and formats the data for use with AWS Glue Studio jobs. You can create connectors for Spark, Athena, and JDBC data stores. Sample code posted on GitHub provides an overview of the basic interfaces you need to implement.

You will need a local development environment for creating your connector code. You can use any IDE or even just a command line editor to write your connector. Examples of development environments include:

- A local Scala environment with a local AWS Glue ETL Maven library, as described in [Developing Locally with Scala](#) in the *AWS Glue Developer Guide*.
- IntelliJ IDE, by downloading the IDE from <https://www.jetbrains.com/idea/>.

Topics

- [Developing Spark connectors](#) (p. 57)
- [Developing Athena connectors](#) (p. 57)
- [Developing JDBC connectors](#) (p. 58)
- [Examples of using custom connectors with AWS Glue Studio](#) (p. 58)
- [Developing AWS Glue connectors for AWS Marketplace](#) (p. 58)

Developing Spark connectors

You can create a Spark connector with Spark DataSource API V2 (Spark 2.4) to read data.

To create a custom Spark connector

Follow the steps in the AWS Glue GitHub sample library for developing Spark connectors, which is located at <https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Spark/README.md>.

Developing Athena connectors

You can create an Athena connector to be used by AWS Glue and AWS Glue Studio to query a custom data source.

To create a custom Athena connector

Follow the steps in the AWS Glue GitHub sample library for developing Athena connectors, which is located at <https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Athena>.

Developing JDBC connectors

You can create a connector that uses JDBC to access your data stores.

To create a custom JDBC connector

1. Install the AWS Glue Spark runtime libraries in your local development environment. Refer to the instructions in the AWS Glue GitHub sample library at <https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/GlueSparkRuntime/README.md>.
2. Implement the JDBC driver that is responsible for retrieving the data from the data source. Refer to the [Java Documentation](#) for Java SE 8.

Create an entry point within your code that AWS Glue Studio uses to locate your connector. The **Class name** field should be the full path of your JDBC driver.

3. Use the `GlueContext` API to read data with the connector. Users can add more input options in the AWS Glue Studio console to configure the connection to the data source, if necessary. For a code example that shows how to read from and write to a JDBC database with a custom JDBC connector, see [Custom and AWS Marketplace connectionType values](#).

Examples of using custom connectors with AWS Glue Studio

You can refer to the following blogs for examples of using custom connectors:

- [Developing, testing, and deploying custom connectors for your data stores with AWS Glue](#)
- Apache Hudi: [Writing to Apache Hudi tables using AWS Glue Custom Connector](#)
- Google BigQuery: [Migrating data from Google BigQuery to Amazon S3 using AWS Glue custom connectors](#)
- Snowflake (JDBC): [Performing data transformations using Snowflake and AWS Glue](#)
- SingleStore: [Building fast ETL using SingleStore and AWS Glue](#)
- Salesforce: [Ingest Salesforce data into Amazon S3 using the CData JDBC custom connector with AWS Glue -](#)
- MongoDB: [Building AWS Glue Spark ETL jobs using Amazon DocumentDB \(with MongoDB compatibility\) and MongoDB](#)
- Amazon Relational Database Service (Amazon RDS): [Building AWS Glue Spark ETL jobs by bringing your own JDBC drivers for Amazon RDS](#)
- MySQL (JDBC): <https://github.com/aws-samples/aws-glue-samples/blob/master/GlueCustomConnectors/development/Spark/SparkConnectorMySQL.scala>

Developing AWS Glue connectors for AWS Marketplace

As an AWS partner, you can create custom connectors and upload them to AWS Marketplace to sell to AWS Glue customers.

The process for developing the connector code is the same as for custom connectors, but the process of uploading and verifying the connector code is more detailed. Refer to the instructions in [Creating Connectors for AWS Marketplace](#) on the GitHub website.

Restrictions for using connectors and connections in AWS Glue Studio

When you're using custom connectors or connectors from AWS Marketplace, take note of the following restrictions:

- The testConnection API isn't supported with connections created for custom connectors.
- Data Catalog connection password encryption isn't supported with custom connectors.
- You can't use job bookmarks if you specify a filter predicate for a data source node that uses a JDBC connector.
- Currently, you can't use custom connectors with data previews.

Tutorial: Using the open-source Elasticsearch Spark Connector

Elasticsearch is a popular open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. You can use OpenSearch as a data store for your extract, transform, and load (ETL) jobs by configuring the Elasticsearch Spark Connector in AWS Glue Studio. This connector is available for free from [AWS Marketplace](#).

In this tutorial, we will show how to connect to your Amazon OpenSearch Service nodes with a minimal number of steps.

Topics

- [Prerequisites \(p. 60\)](#)
- [Step 1: \(Optional\) Create an AWS secret for your OpenSearch cluster information \(p. 60\)](#)
- [Step 2: Subscribe to the connector \(p. 61\)](#)
- [Step 3: Activate the connector in AWS Glue Studio and create a connection \(p. 62\)](#)
- [Step 4: Configure an IAM role for your ETL job \(p. 62\)](#)
- [Step 5: Create a job that uses the OpenSearch connection \(p. 63\)](#)
- [Step 6: Run the job \(p. 65\)](#)

Prerequisites

To use this tutorial, you must have the following:

- Access to AWS Glue Studio
- Access to an OpenSearch cluster in the AWS Cloud
- Configured access to the Amazon VPC that contains your data store, as described in [Configuring a VPC for your ETL job \(p. 9\)](#).
- Configured permissions according to [Overview of Job-related permissions \(p. 8\)](#)
- (Optional) Access to AWS Secrets Manager.

Step 1: (Optional) Create an AWS secret for your OpenSearch cluster information

To safely store and use your connection credential, save your credential in AWS Secrets Manager. The secret you create will be used later in the tutorial by the connection. The credential key-value pairs will be fed into the Elasticsearch Spark Connector as normal connection options.

For more information about creating secrets, see [Creating and Managing Secrets with AWS Secrets Manager](#) in the *AWS Secrets Manager User Guide*.

To create an AWS secret

1. Sign in to the [AWS Secrets Manager console](#).
2. On either the service introduction page or the **Secrets** list page, choose **Store a new secret**.
3. On the **Store a new secret** page, choose **Other type of secret**. This option means that you must supply the structure and details of your secret.
4. Add a **Key** and **Value** pair for the OpenSearch cluster user name. For example:

```
es.net.http.auth.user: username
```

5. Choose **+ Add row**, and enter another key-value pair for the password. For example:

```
es.net.http.auth.pass: password
```

6. Choose **Next**.
7. Enter a secret name. For example: **my-es-secret**. You can optionally include a description.

Record the secret name, which is used later in this tutorial, and then choose **Next**.
8. Choose **Next** again, and then choose **Store** to create the secret.

Next step


[Step 2: Subscribe to the connector \(p. 61\)](#)

Step 2: Subscribe to the connector

The Elasticsearch Spark Connector is available for free from [AWS Marketplace](#).

To subscribe to the Elasticsearch Spark Connector on AWS Marketplace

1. If you have not already configured your AWS account to use License Manager, do the following:
 - a. Open the AWS License Manager console at <https://console.aws.amazon.com/license-manager>.
 - b. Choose **Create customer managed license**.
 - c. In the **IAM permissions (one-time setup)** window, choose **I grant AWS License Manager the required permissions**, and then choose **Grant permissions**.

If you do not see this window, then you have already configured the necessary permissions.
2. Open the AWS Glue Studio console at <https://console.aws.amazon.com/gluestudio/>.
3. In the AWS Glue Studio console, expand the menu icon () , and then choose **Connectors** in the navigation pane.
4. On the **Connectors** page, choose **Go to AWS Marketplace**.
5. In AWS Marketplace, in the **Search AWS Glue Studio products** section, enter **elasticsearch connector** in the search field, and then press Enter.
6. Choose the name of the connector, **ElasticSearch Spark connector for AWS Glue**.
7. On the product page for the connector, use the tabs to view information about the connector. When you're ready to continue, choose **Continue to Subscribe**.
8. Review and accept the terms of use for the software.
9. When the subscription process completes, choose **Continue to Configuration**.
10. Keep the default choices on the **Configure this software** page, and choose **Continue to Launch**.

Next step

[Step 3: Activate the connector in AWS Glue Studio and create a connection \(p. 62\)](#)

Step 3: Activate the connector in AWS Glue Studio and create a connection

After you choose **Continue to Launch**, you see the **Launch this software** page in AWS Marketplace. After you use the link to activate the connector in AWS Glue Studio, you create a connection.

To deploy the connector and create a connection in AWS Glue Studio

1. On the **Launch this software** page in the AWS Marketplace console, choose **Usage Instructions**, and then choose the link in the window that appears.

Your browser is redirected to the AWS Glue Studio console **Create marketplace connection** page.

2. Enter a name for the connection. For example: **my-es-connection**.
3. In the **Connection access** section, for **Connection credential type**, choose **User name and password**.
4. For the **AWS secret**, enter the name of your secret. For example: **my-es-secret**.
5. In the **Network options** section, enter the VPC information to connect to Elastic Search cluster.
6. Choose **Create connection and activate connector**.

Next step

[Step 4: Configure an IAM role for your ETL job \(p. 62\)](#)

Step 4: Configure an IAM role for your ETL job

When you create the AWS Glue ETL job, you specify an AWS Identity and Access Management (IAM) role for the job to use. The role must grant access to all resources used by the job, including Amazon S3 (for any sources, targets, scripts, driver files, and temporary directories), and also AWS Glue Data Catalog objects.

The assumed IAM role for the AWS Glue ETL job must also have access to the secret that was created in the previous section. By default, the AWS managed role `AWSGlueServiceRole` does not have access to the secret. To set up access control for your secrets, see [Authentication and Access Control for AWS Secrets Manager](#) and [Limiting Access to Specific Secrets](#).

To configure an IAM role for your ETL job

1. Configure the permissions described in [the section called "Overview of Job-related permissions" \(p. 8\)](#).
2. Configure the additional permissions needed when using connectors with AWS Glue Studio, as described in [the section called "Permissions required for using connectors" \(p. 9\)](#).

Next step

[Step 5: Create a job that uses the OpenSearch connection \(p. 63\)](#)

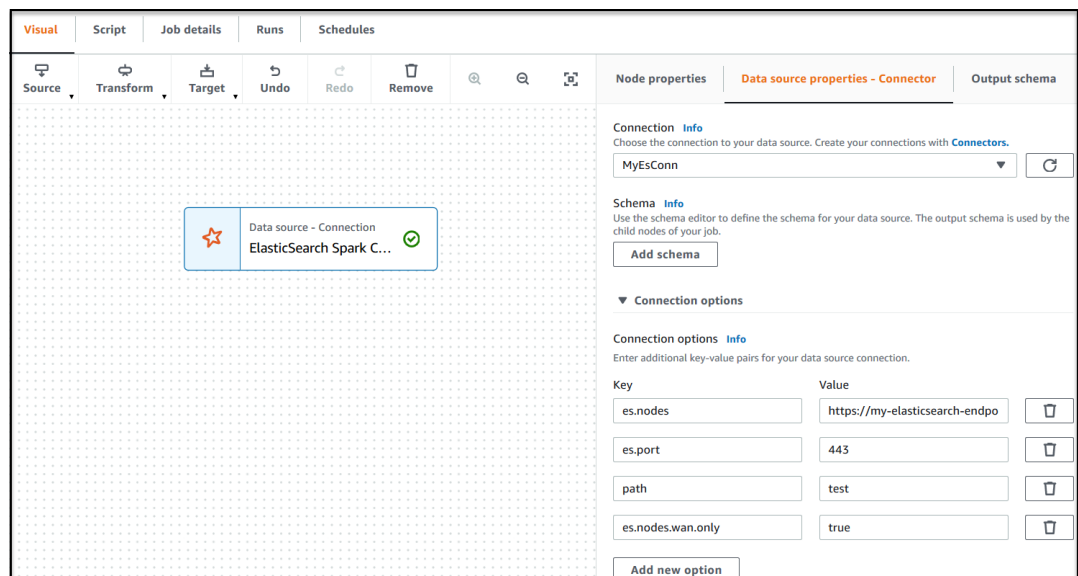
Step 5: Create a job that uses the OpenSearch connection

After creating a role for your ETL job, you can create a job in AWS Glue Studio that uses the connection and connector for Open Spark ElasticSearch.

If your job runs within a Amazon Virtual Private Cloud (Amazon VPC), make sure the VPC is configured correctly. For more information, see [the section called “Configuring a VPC for your ETL job” \(p. 9\)](#).

To create a job that uses the Elasticsearch Spark Connector

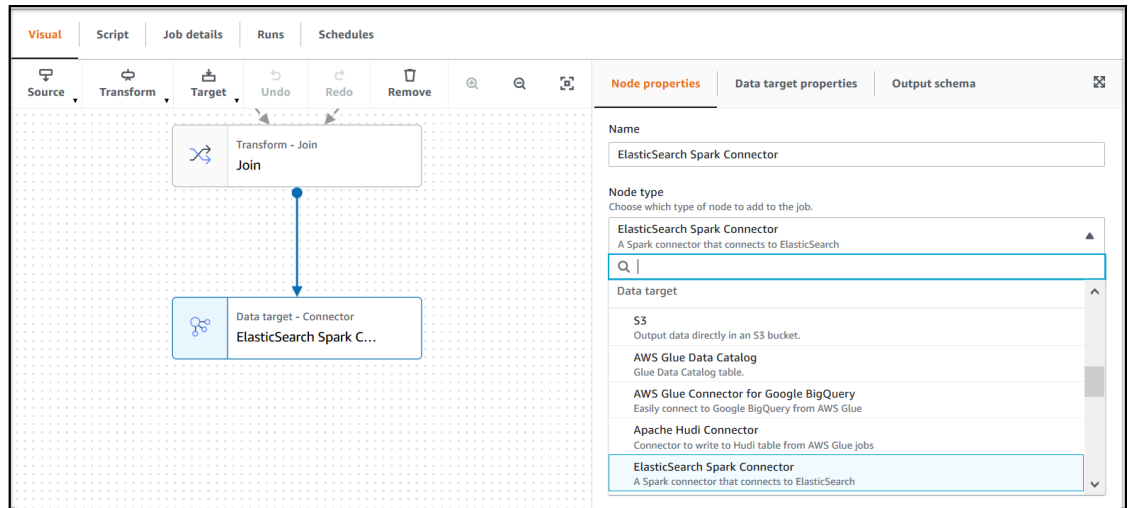
1. In AWS Glue Studio, choose **Connectors**.
2. In the **Your connections** list, select the connection you just created and choose **Create job**.
3. In the visual job editor, choose the Data source node. On the right, on the **Data source properties - Connector** tab, configure additional information for the connector.
 - a. Choose **Add schema** and enter the schema of the data set in the data source. Connections do not use tables stored in the Data Catalog, which means that AWS Glue Studio doesn't know the schema of the data. You must manually provide this schema information. For instructions on how to use the schema editor, see [the section called “Editing the schema in a custom transform node” \(p. 37\)](#).
 - b. Expand **Connection options**.
 - c. Choose **Add new option** and enter the information needed for the connector that was not entered in the AWS secret:
 - **es.nodes** : **https://<ElasticSearch endpoint>**
 - **es.port** : **443**
 - **path** : **test**
 - **es.nodes.wan.only** : **true**



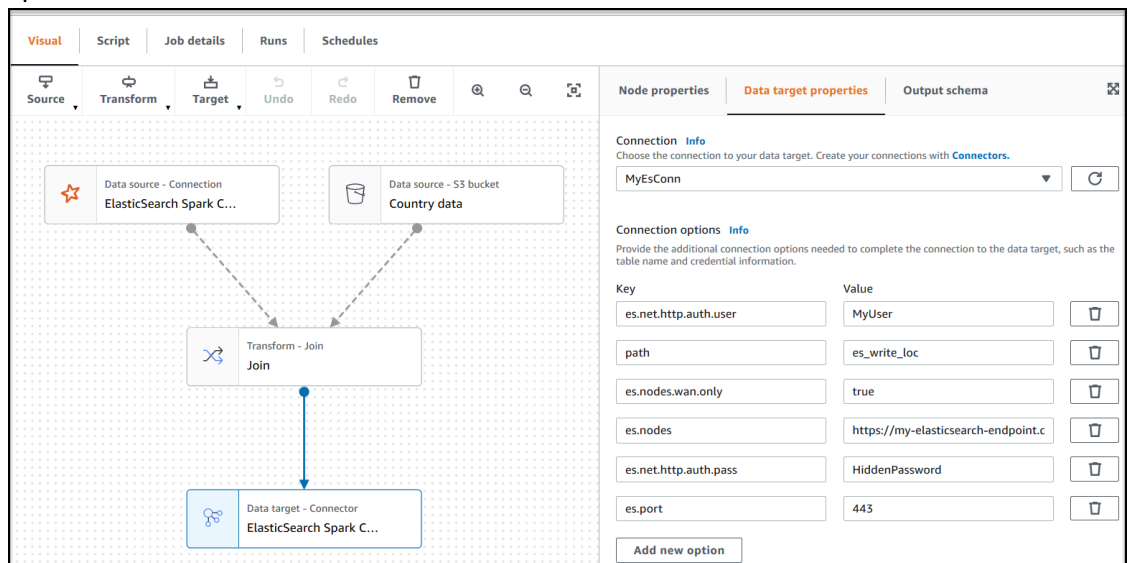
For an explanation of these connection options, refer to: <https://www.elastic.co/guide/en/elasticsearch/hadoop/current/configuration.html>.

4. Add a target node to the graph as described in [the section called “Adding nodes to the job diagram” \(p. 44\)](#) and [the section called “Editing the data target node” \(p. 39\)](#).

Your data target can be Amazon S3, or it can use information from an AWS Glue Data Catalog or a connector to write data in a different location. For example, you can use a Data Catalog table to write to a database in Amazon RDS, or you can use a connector as your data target to write to data stores that are not natively supported in AWS Glue.



If you choose a connector for your data target, you must choose a connection created for that connector. Also, if required by the connector provider, you must add options to provide additional information to the connector. If you use a connection that contains information for an AWS secret, then you don't need to provide the user name and password authentication in the connection options.



5. Optionally, add additional data sources and one or more transform nodes as described in [the section called “Editing the data transform node” \(p. 25\)](#).
6. Configure the job properties as described in [the section called “Modify the job properties” \(p. 69\)](#), starting with step 3, and save the job.

Next step

[Step 6: Run the job \(p. 65\)](#)

Step 6: Run the job

After you save your job, you can run the job to perform the ETL operations.

To run the job you created for the Elasticsearch Spark Connector

1. Using the AWS Glue Studio console, on the visual editor page, choose **Run**.
2. In the success banner, choose **Run Details**, or you can choose the **Runs** tab of the visual editor to view information about the job run.

Managing ETL jobs with AWS Glue Studio

You can use the simple graphical interface in AWS Glue Studio to manage your ETL jobs. Using the navigation menu, choose **Jobs** to view the **Jobs** page. On this page, you can see all the jobs that you have created either with AWS Glue Studio or the AWS Glue console. You can view, manage, and run your jobs on this page.

You can also perform the following tasks on this page:

- [Start a job run](#) (p. 66)
- [Schedule job runs](#) (p. 66)
- [Manage job schedules](#) (p. 67)
- [Stop job runs](#) (p. 68)
- [View your jobs](#) (p. 68)
- [View information for recent job runs](#) (p. 68)
- [View the job script](#) (p. 69)
- [Modify the job properties](#) (p. 69)
- [Save the job](#) (p. 71)
- [Clone a job](#) (p. 73)
- [Delete jobs](#) (p. 73)

Start a job run

In AWS Glue Studio, you can run your jobs on demand. A job can run multiple times, and each time you run the job, AWS Glue collects information about the job activities and performance. This information is referred to as a *job run* and is identified by a job run ID.

You can initiate a job run in the following ways in AWS Glue Studio:

- On the **Jobs** page, choose the job you want to start, and then choose the **Run job** button.
- If you're viewing a job in the visual editor and the job has been saved, you can choose the **Run** button to start a job run.

For more information about job runs, see [Working with Jobs on the AWS Glue Console](#) in the *AWS Glue Developer Guide*.

Schedule job runs

In AWS Glue Studio, you can create a schedule to have your jobs run at specific times. You can specify constraints, such as the number of times that the jobs run, which days of the week they run, and at what time. These constraints are based on `cron` and have the same limitations as `cron`. For example, if you choose to run your job on day 31 of each month, keep in mind that some months don't have 31 days. For more information about `cron`, see [Cron Expressions](#) in the *AWS Glue Developer Guide*.

To run jobs according to a schedule

1. Create a job schedule using one of the following methods:

- On the **Jobs** page, choose the job you want to create a schedule for, choose **Actions**, and then choose **Schedule job**.
 - If you're viewing a job in the visual editor and the job has been saved, choose the **Schedules** tab. Then choose **Create Schedule**.
2. On the **Schedule job run** page, enter the following information:
 - **Name:** Enter a name for your job schedule.
 - **Frequency:** Enter the frequency for the job schedule. You can choose the following:
 - **Hourly:** The job will run every hour, starting at a specific minute. You can specify the **Minute** of the hour that the job should run. By default, when you choose hourly, the job runs at the beginning of the hour (minute 0).
 - **Daily:** The job will run every day, starting at a time. You can specify the **Minute** of the hour that the job should run and the **Start hour** for the job. Hours are specified using a 23-hour clock, where you use the numbers 13 to 23 for the afternoon hours. The default values for minute and hour are 0, which means that if you select **Daily**, the job by default will run at midnight.
 - **Weekly:** The job will run every week on one or more days. In addition to the same settings described previous for Daily, you can choose the days of the week on which the job will run. You can choose one or more days.
 - **Monthly:** The job will run every month on a specific day. In addition to the same settings described previous for Daily, you can choose the day of the month on which the job will run. Specify the day as a numeric value from 1 to 31. If you select a day that does not exist in a month, for example the 30th day of February, then the job does not run that month.
 - **Custom:** Enter an expression for your job schedule using the `cron` syntax. Cron expressions allow you to create more complicated schedules, such as the last day of the month (instead of a specific day of the month) or every third month on the 7th and 21st days of the month.

See [Cron Expressions](#) in the *AWS Glue Developer Guide*

 - **Description:** You can optionally enter a description for your job schedule. If you plan to use the same schedule for multiple jobs, having a description makes it easier to determine what the job schedule does.
3. Choose **Create schedule** to save the job schedule.
 4. After you create the schedule, a success message appears at the top of the console page. You can choose **Job details** in this banner to view the job details. This opens the visual job editor page, with the **Schedules** tab selected.

Manage job schedules

After you have created schedules for a job, you can open the job in the visual editor and choose the **Schedules** tab to manage the schedules.

On the **Schedules** tab of the visual editor, you can perform the following tasks:

- Create a new schedule.

Choose **Create schedule**, then enter the information for your schedule as described in [the section called "Schedule job runs" \(p. 66\)](#).

- Edit an existing schedule.

Choose the schedule you want to edit, then choose **Action** followed by **Edit schedule**. When you choose to edit an existing schedule, the **Frequency** shows as **Custom**, and the schedule is displayed as a `cron` expression. You can either modify the `cron` expression, or specify a new schedule using the **Frequency** button. When you finish with your changes, choose **Update schedule**.

- Pause an active schedule.

Choose an active schedule, and then choose **Action** followed by **Pause schedule**. The schedule is instantly deactivated. Choose the refresh (reload) button to see the updated job schedule status.

- Resume a paused schedule.

Choose a deactivated schedule, and then choose **Action** followed by **Resume schedule**. The schedule is instantly activated. Choose the refresh (reload) button to see the updated job schedule status.

- Delete a schedule.

Choose the schedule you want to remove, and then choose **Action** followed by **Delete schedule**. The schedule is instantly deleted. Choose the refresh (reload) button to see the updated job schedule list. The schedule will show a status of **Deleting** until it has been completely removed.

Stop job runs

You can stop a job before it has completed its job run. You might choose this option if you know that the job isn't configured correctly, or if the job is taking too long to complete.

On the **Monitoring** page, in the **Job runs** list, choose the job that you want to stop, choose **Actions**, and then choose **Stop run**.

View your jobs


You can view all your jobs on the **Jobs** page. You can access this page by choosing **Jobs** in the navigation pane.

On the **Jobs** page, you can see all the jobs that were created in your account. The **Your jobs** list shows the job name, its type, the status of the last run of that job, and the dates on which the job was created and last modified. You can choose the name of a job to see detailed information for that job.

You can also use the Monitoring dashboard to view all your jobs. You can access the dashboard by choosing **Monitoring** in the navigation pane. For more information about using the dashboard, see [Monitoring ETL jobs in AWS Glue Studio \(p. 74\)](#).

Customize the job display

You can customize how the jobs are displayed in the **Your jobs** section of the **Jobs** page. Also, you can enter text in the search text field to display only jobs with a name that contains that text.

If you choose the settings icon  in the **Your jobs** section, you can customize how AWS Glue Studio displays the information in the table. You can choose to wrap the lines of text in the display, change the number of jobs displayed on the page, and specify which columns to display.

View information for recent job runs

A job can run multiple times as new data is added at the source location. Each time a job runs, the job run is assigned a unique ID, and information about that job run is collected. You can view this information by using the following methods:

- Choose the **Runs** tab of the visual editor to view the job run information for the currently displayed job.

On the **Runs** tab (the **Recent job runs** page), there is a card for each job run. The information displayed on the **Runs** tab includes:

- Job run ID
 - Number of attempts to run this job
 - Status of the job run
 - Start and end time for the job run
 - The runtime for the job run
 - A link to the job log files
 - A link to the job error log files
 - The error returned for failed jobs
- In the navigation pane, choose **Monitoring**. Scroll down to the **Job runs** list. Choose the job and then choose **View run details**.

The information displayed on the job run detail page accessed from the **Monitoring** page is more comprehensive. The contents are described in [Viewing the details of a job run \(p. 76\)](#).

For more information about the job logs, see [Viewing the job run logs \(p. 76\)](#).

View the job script

After you provide information for all the nodes in the job, AWS Glue Studio generates a script that is used by the job to read the data from the source, transform the data, and write the data in the target location. If you save the job, you can view this script at any time.

To view the generated script for your job

1. Choose **Jobs** in the navigation pane.
2. On the **Jobs** page, in the **Your Jobs** list, choose the name of the job you want to review. Alternatively, you can select a job in the list, choose the **Actions** menu, and then choose **Edit job**.
3. On the visual editor page, choose the **Script** tab at the top to view the job script.

If you want to edit the job script, see [Editing or uploading a job script \(p. 41\)](#).

Modify the job properties

The nodes in the job diagram define the actions performed by the job, but there are several properties that you can configure for the job as well. These properties determine the environment that the job runs in, the resources it uses, the threshold settings, the security settings, and more.

To customize the job run environment

1. Choose **Jobs** in the navigation pane.
2. On the **Jobs** page, in the **Your Jobs** list, choose the name of the job you want to review.
3. On the visual editor page, choose the **Job details** tab at the top of the job editing pane.
4. Modify the job properties, as needed.

For more information about the job properties, see [Defining Job Properties](#) in the *AWS Glue Developer Guide*.

5. Expand the **Advanced properties** section if you need to specify these additional job properties:

- **Script filename** – The name of the file that stores the job script in Amazon S3.
 - **Script path** – The Amazon S3 location where the job script is stored.
 - **Job metrics** – (Not available for Python shell jobs) Turns on the creation of Amazon CloudWatch metrics when this job runs.
 - **Continuous logging** – (Not available for Python shell jobs) Turns on continuous logging to CloudWatch, so the logs are available for viewing before the job completes.
 - **Spark UI** and **Spark UI logs path** – (Not available for Python shell jobs) Turns on the use of Spark UI for monitoring this job and specifies the location for the Spark UI logs.
 - **Maximum concurrency** – Sets the maximum number of concurrent runs that are allowed for this job.
 - **Temporary path** – The location of a working directory in Amazon S3 where temporary intermediate results are written when AWS Glue runs the job script.
 - **Delay notification threshold (minutes)** – Specify a delay threshold for the job. If the job runs for a longer time than that specified by the threshold, then AWS Glue sends a delay notification for the job to CloudWatch.
 - **Security configuration** and **Server-side encryption** – Use these fields to choose the encryption options for the job.
 - **Use Glue Data Catalog as the Hive metastore** – Choose this option if you want to use the AWS Glue Data Catalog as an alternative to Apache Hive Metastore.
 - **Additional network connection** – For a data source in a VPC, you can specify a connection of type *Network* to ensure your job accesses your data through the VPC.
 - **Python library path**, **Dependent jars path** (Not available for Python shell jobs), or **Referenced files path** – Use these fields to specify the location of additional files used by the job when it runs the script.
 - **Job Parameters** – You can add a set of key-value pairs that are passed as named parameters to the job script. In Python calls to AWS Glue APIs, it's best to pass parameters explicitly by name. For more information about using parameters in a job script, see [Passing and Accessing Python Parameters in AWS Glue](#) in the *AWS Glue Developer Guide*.
 - **Tags** – You can add tags to the job to help you organize and identify them.
6. After you have modified the job properties, save the job.

Store Spark shuffle files on Amazon S3

Some ETL jobs require reading and combining information from multiple partitions, for example, when using a join transform. This operation is referred to as *shuffling*. During a shuffle, data is written to disk and transferred across the network. With AWS Glue version 3.0, you can configure Amazon S3 as a storage location for these files. AWS Glue provides a shuffle manager which writes and reads shuffle files to and from Amazon S3. Writing and reading shuffle files from Amazon S3 is slower (by 5%-20%) compared to local disk (or Amazon EBS which is heavily optimized for Amazon EC2). However, Amazon S3 provides unlimited storage capacity, so you don't have to worry about "No space left on device" errors when running your job.

To configure your job to use Amazon S3 for shuffle files

1. On the **Jobs** page, in the **Your Jobs** list, choose the name of the job you want to modify.
2. On the visual editor page, choose the **Job details** tab at the top of the job editing pane.

Scroll down to the **Job parameters** section.

3. Specify the following key-value pairs.
 - `--write-shuffle-files-to-s3 — true`

This is the main parameter that configures the shuffle manager in AWS Glue to use Amazon S3 buckets for writing and reading shuffle data. By default, this parameter has a value of `false`.

- (Optional) `--write-shuffle-spills-to-s3 — true`

This parameter allows you to offload spill files to Amazon S3 buckets, which provides additional resiliency to your Spark job in AWS Glue. This is only required for large workloads that spill a lot of data to disk. By default, this parameter has a value of `false`.

- (Optional) `--conf spark.shuffle.glue.s3ShuffleBucket — S3://<shuffle-bucket>`

This parameter specifies the Amazon S3 bucket to use when writing the shuffle files. If you do not set this parameter, the location is the `shuffle-data` folder in the location specified for **Temporary path** (`--TempDir`).

Note

Make sure the location of the shuffle bucket is in the same AWS Region in which the job runs.

Also, the shuffle service does not clean the files after the job finishes running, so you should configure the Amazon S3 storage life cycle policies on the shuffle bucket location. For more information, see [Managing your storage lifecycle](#) in the *Amazon S3 User Guide*.

Save the job

A red **Job has not been saved** callout is displayed to the left of the **Save** button until you save the job.



To save your job

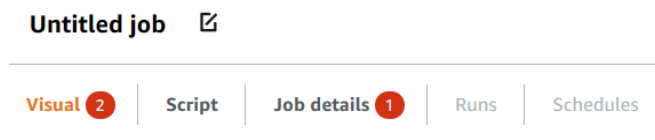
1. Provide all the required information in the **Visual** and **Job details** tabs.
2. Choose the **Save** button.


After you save the job, the 'not saved' callout changes to display the time and date that the job was last saved.

If you exit AWS Glue Studio before saving your job, the next time you sign in to AWS Glue Studio, a notification appears. The notification indicates that there is an unsaved job, and asks if you want to restore it. If you choose to restore the job, you can continue to edit it.

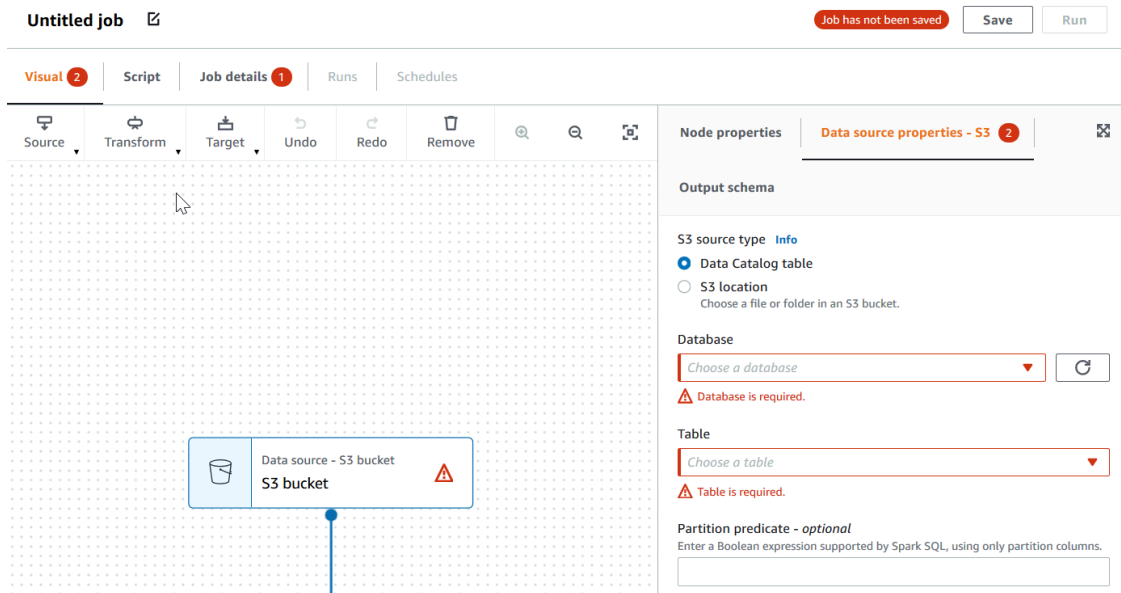
Troubleshooting errors when saving a job

If you choose the **Save** button, but your job is missing some required information, then a red callout appears on the tab where the information is missing. The number in the callout indicates how many missing fields were detected.



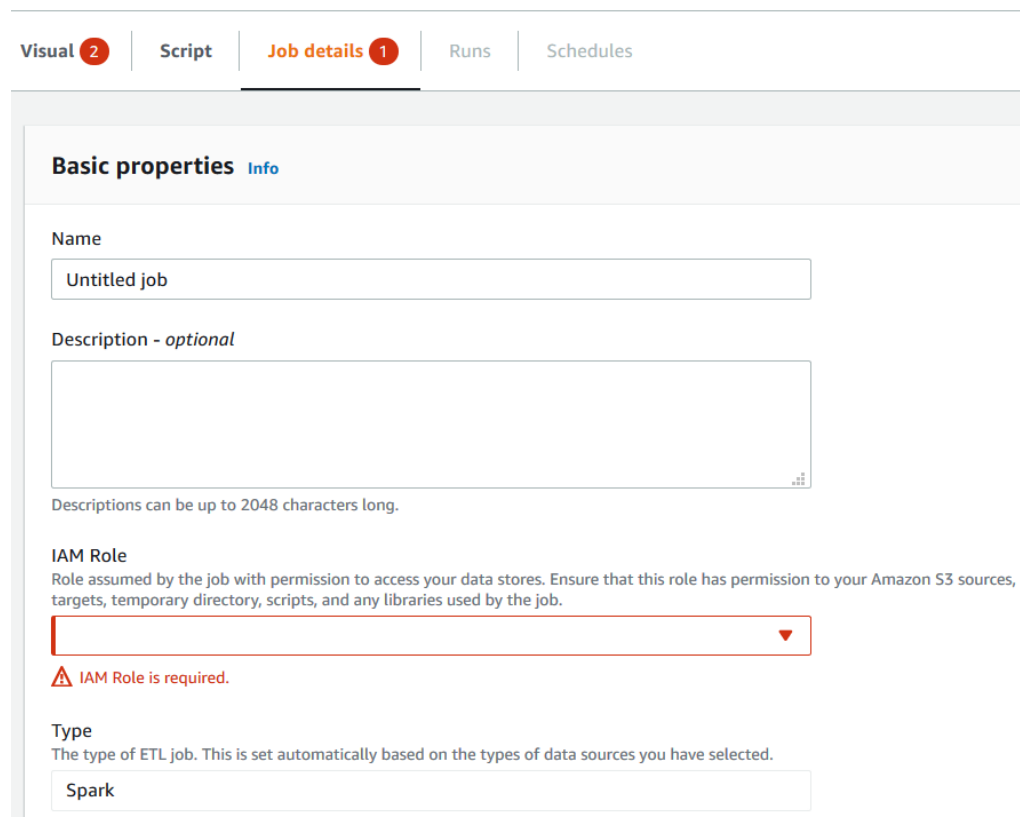
- If a node in the visual editor isn't configured correctly, the **Visual** tab shows a red callout, and the node with the error displays a warning symbol .
- 1. Choose the node. In the node details panel, a red callout appears on the tab where the missing or incorrect information is located.

2. Choose the tab in the node details panel that shows a red callout, and then locate the problem fields, which are highlighted. An error message below the fields provides additional information about the problem.



- If there is a problem with the job properties, the **Job details** tab shows a red callout. Choose that tab and locate the problem fields, which are highlighted. The error messages below the fields provide additional information about the problem.

Untitled job



Clone a job

You can use the **Clone job** action to copy an existing job into a new job.

To create a new job by copying an existing job

1. On the **Jobs** page, in the **Your jobs** list, choose the job that you want to duplicate.
2. From the **Actions** menu, choose **Clone job**.
3. Enter a name for the new job. You can then save or edit the job.

Delete jobs

You can remove jobs that are no longer needed. You can delete one or more jobs in a single operation.

To remove jobs from AWS Glue Studio

1. On the **Jobs** page, in the **Your jobs** list, choose the jobs that you want to delete.
2. From the **Actions** menu, choose **Delete job**.
3. Verify that you want to delete the job by entering **delete**.

You can also delete a saved job when you're viewing the **Job details** tab for that job in the visual editor.

Monitoring ETL jobs in AWS Glue Studio

Monitoring is an important part of maintaining the reliability, availability, and performance of ETL jobs used in AWS Glue and AWS Glue Studio. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs.

Topics

- [Accessing the job monitoring dashboard \(p. 74\)](#)
- [Overview of the job monitoring dashboard \(p. 74\)](#)
- [Job runs view \(p. 74\)](#)
- [Viewing the job run logs \(p. 76\)](#)
- [Viewing the details of a job run \(p. 76\)](#)
- [Viewing Amazon CloudWatch metrics for a job run \(p. 77\)](#)

Accessing the job monitoring dashboard

You access the job monitoring dashboard by choosing the **Monitoring** link in the AWS Glue Studio navigation pane.

Overview of the job monitoring dashboard

The job monitoring dashboard provides an overall summary of the job runs, with totals for the jobs with a status of **Running**, **Canceled**, **Success**, or **Failed**. Additional tiles provide the overall job run success rate, the estimated DPU usage for jobs, a breakdown of the job status counts by job type, worker type, and by day.

The graphs in the tiles are interactive. You can choose any block in a graph to run a filter that displays only those jobs in the **Job runs** table at the bottom of the page.

You can change the date range for the information displayed on this page by using the **Date range** selector. When you change the date range, the information tiles adjust to show the values for the specified number of days before the current date. You can also use a specific date range if you choose **Custom** from the date range selector.

Job runs view

The **Job runs** resource list shows the jobs for the specified date range and filters.

You can filter the jobs on additional criteria, such as status, worker type, job type, and the job name. In the filter box at the top of the table, you can enter the text to use as a filter. The table results are updated with rows that contain matching text as you enter the text.

You can view a subset of the jobs by choosing elements from the graphs on the job monitoring dashboard. For example, if you choose the number of running jobs in the **Job runs summary** tile, then

the **Job runs** list displays only the jobs that currently have a status of **Running**. If you choose one of the bars in the **Worker type breakdown** bar chart, then only job runs with the matching worker type and status are shown in the **Job runs** list.

The **Job runs** resource list displays the details for the job runs. You can sort the rows in the table by choosing a column heading. The table contains the following information:

Property	Description
Job name	The name of the job
Type	The type of job environment: <ul style="list-style-type: none"> • Glue ETL: Runs in an Apache Spark environment managed by AWS Glue. • Glue Streaming: Runs in an Apache Spark environment and performs ETL on data streams. • Python shell: Runs Python scripts as a shell
Start time	The date and time at which this job run was started.
End time	The date and time that this job run completed.
Run status	The current state of the job run. Values can be: <ul style="list-style-type: none"> • STARTING • RUNNING • STOPPING • STOPPED • SUCCEEDED • FAILED • TIMEOUT
Run time	The amount of time that the job run consumed resources.
Capacity	The number of AWS Glue data processing units (DPUs) that were allocated for this job run. For more information about capacity planning, see Monitoring for DPU Capacity Planning in the <i>AWS Glue Developer Guide</i> .
Worker type	The type of predefined worker that was allocated when the job ran. Values can be Standard , G.1X , or G.2X .
DPU hours	The estimated number of DPUs used for the job run. A DPU is a relative measure of processing power. DPUs are used to determine the cost of running your job. For more information, see the AWS Glue pricing page.

You can choose any job run in the list and view additional information. Choose a job run, and then do one of the following:

- Choose the **Actions** menu and the **View job** option to view the job in the visual editor.
- Choose the **Actions** menu and the **Stop run** option to stop the current run of the job.
- Choose the **View CloudWatch logs** button to view the job run logs for that job.
- Choose **View run details** to view the job run details page.

Viewing the job run logs

You can view the job logs in a variety of ways:

- On the **Monitoring** page, in the **Job runs** table, choose a job run, and then choose **View CloudWatch logs**.
- In the visual job editor, on the **Runs** tab for a job, choose the hyperlinks to view the logs:
 - **Logs** – Links to the Apache Spark job logs written when continuous logging is enabled for a job run. When you choose this link, it takes you to the Amazon CloudWatch logs in the `/aws-glue/jobs/logs-v2` log group. By default, the logs exclude non-useful Apache Hadoop YARN heartbeat and Apache Spark driver or executor log messages. For more information about continuous logging, see [Continuous Logging for AWS Glue Jobs](#) in the *AWS Glue Developer Guide*.
 - **Error logs** – Links to the logs written to `stderr` for this job run. When you choose this link, it takes you to the Amazon CloudWatch logs in the `/aws-glue/jobs/error` log group. You can use these logs to view details about any errors that were encountered during the job run.
 - **Output logs** – Links to the logs written to `stdout` for this job run. When you choose this link, it takes you to the Amazon CloudWatch logs in the `/aws-glue/jobs/output` log group. You can use these logs to see all the details about the tables that were created in the AWS Glue Data Catalog and any errors that were encountered.

Viewing the details of a job run

You can choose a job in the **Job runs** list on the **Monitoring** page, and then choose **View run details** to see detailed information for that run of the job.

The information displayed on the job run detail page includes:

Property	Description
Job name	The name of the job
Run Status	The current state of the job run. Values can be: <ul style="list-style-type: none">• <code>STARTING</code>• <code>RUNNING</code>• <code>STOPPING</code>• <code>STOPPED</code>• <code>SUCCEEDED</code>• <code>FAILED</code>• <code>TIMEOUT</code>
Glue version	The AWS Glue version used by the job run
Recent attempt	The number of automatic retry attempts for this job run

Property	Description
Start time	The date and time at which this job run was started
End time	The date and time that this job run completed
Start-up time	The amount of time spent preparing to run the job
Execution time	The amount of time spent running the job script
Trigger name	The name of the trigger associated with the job
Last modified on	The date when the job was last modified
Security configuration	The security configuration for the job, which includes Amazon S3 encryption, CloudWatch encryption, and job bookmarks encryption settings
Timeout	The job run timeout threshold value
Allocated capacity	The number of AWS Glue data processing units (DPUs) that were allocated for this job run. For more information about capacity planning, see Monitoring for DPU Capacity Planning in the <i>AWS Glue Developer Guide</i> .
Max capacity	The maximum capacity available to the job run.
Number of workers	The number of workers used for the job run
Worker type	The type of predefined workers allocated for the job run. Values can be <code>Standard</code> , <code>G.1X</code> , or <code>G.2X</code> .
Logs	A link to the job logs for continuous logging (<code>/aws-glue/jobs/logs-v2</code>)
Output Logs	A link to the job output log files (<code>/aws-glue/jobs/output</code>)
Error logs	A link to the job error log files (<code>/aws-glue/jobs/error</code>)

Viewing Amazon CloudWatch metrics for a job run

On the details page for a job run, below the **Run details** section, you can view the job metrics. AWS Glue Studio sends job metrics to Amazon CloudWatch for every job run.

AWS Glue reports metrics to Amazon CloudWatch every 30 seconds. The AWS Glue metrics represent delta values from the previously reported values. Where appropriate, metrics dashboards aggregate (sum) the 30-second values to obtain a value for the entire last minute. However, the Apache Spark metrics that AWS Glue passes on to Amazon CloudWatch are generally absolute values that represent the current state at the time they are reported.

Note

You must configure your account to access Amazon CloudWatch, as described in [Amazon CloudWatch permissions \(p. 7\)](#).

The metrics provide information about your job run, such as:

- **ETL Data Movement** – The number of bytes read from or written to Amazon S3.
- **Memory Profile: Heap used** – The number of memory bytes used by the Java virtual machine (JVM) heap.
- **Memory Profile: heap usage** – The fraction of memory (scale: 0–1), shown as a percentage, used by the JVM heap.
- **CPU Load** – The fraction of CPU system load used (scale: 0–1), shown as a percentage.

Tutorial: Adding an AWS Glue crawler

For this AWS Glue scenario, you're asked to analyze arrival data for major air carriers to calculate the popularity of departure airports month over month. You have flights data for the year 2016 in CSV format stored in Amazon S3. Before you transform and analyze your data, you catalog its metadata in the AWS Glue Data Catalog.

In this tutorial, let's add a crawler that infers metadata from these flight logs in Amazon S3 and creates a table in your Data Catalog.

Topics

- [Prerequisites \(p. 79\)](#)
- [Step 1: Add a crawler \(p. 79\)](#)
- [Step 2: Run the crawler \(p. 80\)](#)
- [Step 3: View AWS Glue Data Catalog objects \(p. 80\)](#)

Prerequisites

This tutorial assumes that you have an AWS account and access to AWS Glue.

Step 1: Add a crawler

Use these steps to configure and run a crawler that extracts the metadata from a CSV file stored in Amazon S3.

To create a crawler that reads files stored on Amazon S3

1. On the AWS Glue service console, on the left-side menu, choose **Crawlers**.
2. On the Crawlers page, choose **Add crawler**. This starts a series of pages that prompt you for the crawler details.
3. In the Crawler name field, enter **Flights Data Crawler**, and choose **Next**.

Crawlers invoke classifiers to infer the schema of your data. This tutorial uses the built-in classifier for CSV by default.

4. For the crawler source type, choose **Data stores** and choose **Next**.
5. Now let's point the crawler to your data. On the **Add a data store** page, choose the Amazon S3 data store. This tutorial doesn't use a connection, so leave the **Connection** field blank if it's visible.

For the option **Crawl data in**, choose **Specified path in another account**. Then, for the **Include path**, enter the path where the crawler can find the flights data, which is **s3://crawler-public-us-east-1/flight/2016/csv**. After you enter the path, the title of this field changes to **Include path**. Choose **Next**.

6. You can crawl multiple data stores with a single crawler. However, in this tutorial, we're using only a single data store, so choose **No**, and then choose **Next**.

7. The crawler needs permissions to access the data store and create objects in the AWS Glue Data Catalog. To configure these permissions, choose **Create an IAM role**. The IAM role name starts with `AWSGlueServiceRole-`, and in the field, you enter the last part of the role name. Enter **CrawlerTutorial**, and then choose **Next**.

Note

To create an IAM role, your AWS user must have `CreateRole`, `CreatePolicy`, and `AttachRolePolicy` permissions.

The wizard creates an IAM role named `AWSGlueServiceRole-CrawlerTutorial`, attaches the AWS managed policy `AWSGlueServiceRole` to this role, and adds an inline policy that allows read access to the Amazon S3 location `s3://crawler-public-us-east-1/flight/2016/csv`.

8. Create a schedule for the crawler. For **Frequency**, choose **Run on demand**, and then choose **Next**.
9. Crawlers create tables in your Data Catalog. Tables are contained in a database in the Data Catalog. First, choose **Add database** to create a database. In the pop-up window, enter **test-flights-db** for the database name, and then choose **Create**.

Next, enter **flights** for **Prefix added to tables**. Use the default values for the rest of the options, and choose **Next**.

10. Verify the choices you made in the **Add crawler** wizard. If you see any mistakes, you can choose **Back** to return to previous pages and make changes.

After you have reviewed the information, choose **Finish** to create the crawler.

Step 2: Run the crawler

After creating a crawler, the wizard sends you to the Crawlers view page. Because you create the crawler with an on-demand schedule, you're given the option to run the crawler.

To run the crawler

1. The banner near the top of this page lets you know that the crawler was created, and asks if you want to run it now. Choose **Run it now?** to run the crawler.

The banner changes to show "Attempting to run" and "Running" messages for your crawler. After the crawler starts running, the banner disappears, and the crawler display is updated to show a status of Starting for your crawler. After a minute, you can click the Refresh icon to update the status of the crawler that is displayed in the table.

2. When the crawler completes, a new banner appears that describes the changes made by the crawler. You can choose the **test-flights-db** link to view the Data Catalog objects.

Step 3: View AWS Glue Data Catalog objects

The crawler reads data at the source location and creates tables in the Data Catalog. A table is the metadata definition that represents your data, including its schema. The tables in the Data Catalog do not contain data. Instead, you use these tables as a source or target in a job definition.

To view the Data Catalog objects created by the crawler

1. In the left-side navigation, under **Data catalog**, choose **Databases**. Here you can view the `flights-db` database that is created by the crawler.
2. In the left-side navigation, under **Data catalog** and below **Databases**, choose **Tables**. Here you can view the `flightscsv` table created by the crawler. If you choose the table name, then you can view

the table settings, parameters, and properties. Scrolling down in this view, you can view the schema, which is information about the columns and data types of the table.

3. If you choose **View partitions** on the table view page, you can see the partitions created for the data. The first column is the partition key.

Document history for AWS Glue Studio User Guide

Latest documentation update: October 5, 2021

The following table describes the important changes in each revision of the *AWS Glue Studio User Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
Direct access to streaming sources now available (p. 82)	When adding data sources to your ETL job in the visual editor, you can supply information to access the data stream instead of having to use a Data Catalog database and table.	September 30, 2021
AWS Glue Studio supports AWS Glue version 3.0 (p. 82)	When creating jobs in AWS Glue Studio, you can choose Glue 3.0 as the version for your job in the Job details tab. If you do not choose a version for your ETL job, Glue 2.0 is used by default.	August 18, 2021
AWS GovCloud (US) Region (p. 82)	AWS Glue Studio is now available in the AWS GovCloud (US) Region	August 18, 2021
Python shell authoring available in AWS Glue Studio (p. 82)	When creating a new job, you can now choose to create a Python shell job. For more information, see Start the job creation process and Editing Python shell jobs in AWS Glue Studio .	August 13, 2021
Upload scripts to AWS Glue Studio (p. 82)	In conjunction with the script editor feature, you can upload job scripts to AWS Glue Studio. For more information, see Start the job creation process and Editing or uploading a job script .	June 14, 2021
View your job's dataset while creating and editing jobs (p. 82)	You can use the new Data preview tab for a node in your job diagram to see a sample of the data processed by that node. For more information, see Using data previews in the visual job editor .	June 7, 2021
Specify settings for your streaming ETL job in the visual job editor (p. 82)	You can configure additional connection settings for streaming data sources in the	June 4, 2021

	visual job editor to optimize your streaming ETL jobs. For more information, see Using a streaming data source .	
Network connection support added (p. 82)	If you want to access a data source located in your VPC, you can specify a network connection for the job. For more information, see Modify the job properties .	May 24, 2021
Edit job scripts (p. 82)	You can now edit scripts in the job editor. For more information, see Editing a job script .	May 24, 2021
Delete jobs using the AWS Glue Studio console (p. 82)	You can now delete jobs in AWS Glue Studio. To learn how, see Delete jobs .	May 24, 2021
Read data from files in child folders in Amazon S3 (p. 82)	You can specify a single folder in Amazon S3 as your data source and use the Recursive option to include all the child folders as part of the data source. For more information, see Using files in Amazon S3 for the data source .	April 30, 2021
Delete connectors and connections functionality added (p. 82)	You can now delete connectors and connections in AWS Glue Studio. For more information, see Deleting connectors and connections .	April 30, 2021
Fill missing values transform added (p. 82)	You can use the <i>FillMissingValues</i> transform in AWS Glue Studio to locate records in the dataset that have missing values and add a new field with an estimated value. For more information, see Editing the data transform node .	March 29, 2021
SQL transform available (p. 82)	You can use a SQL transform node to write your own transform in the form of a SQL query. For more information, see Using a SQL query to transform data .	March 23, 2021
JDBC source nodes now support job bookmark keys (p. 82)	Job bookmarks help AWS Glue maintain state information and prevent the reprocessing of old data. For more information, see Authoring jobs with custom connectors .	March 15, 2021

Connectors can be used for data targets (p. 82)	Using a custom or AWS Marketplace connector for your data target is now supported. For more information, see Authoring jobs with custom connectors .	March 15, 2021
A new toolbar is available for the visual job editor (p. 82)	A more streamlined and functional toolbar is available for the visual job editor of AWS Glue Studio. This feature makes it easier to add nodes to your graph.	March 8, 2021
Read data from Amazon S3 without creating Data Catalog tables (p. 82)	AWS Glue Studio now allows you to read data directly from Amazon S3 without first creating a table in the AWS Glue Data Catalog. For more information, see Editing the data source node .	February 5, 2021
AWS Glue Studio jobs can now update Data Catalog tables (p. 82)	AWS Glue Studio now supports updating the AWS Glue Data Catalog during job runs. This feature makes it easy to keep your tables up to date as your jobs write new data into Amazon S3. This makes the data immediately available for query from any analytics service that is compatible with the AWS Glue Data Catalog. For more information, see Configuring data target nodes .	February 5, 2021
Job scheduling now available in AWS Glue Studio (p. 82)	You can define a time-based schedule for your job runs in AWS Glue Studio. You can use the console to create a basic schedule, or define a more complex schedule using the Unix-like cron syntax. For more information, see Schedule job runs .	December 21, 2020
AWS Glue Custom Connectors released (p. 82)	AWS Glue Custom Connectors allow you to discover and subscribe to connectors in AWS Marketplace. We also released AWS Glue Spark runtime interfaces to plug in connectors built for Apache Spark Datasource, Athena federated query, and JDBC APIs. For more information, see Using Connectors and connections with AWS Glue Studio .	December 21, 2020

Support for running streaming ETL jobs in AWS Glue version 2.0 (p. 82)	AWS Glue Studio now supports running streaming ETL jobs using AWS Glue version 2.0. For more information, see Adding Streaming ETL Jobs in AWS Glue in the <i>AWS Glue Developer Guide</i> .	November 11, 2020
Availability of AWS Glue Studio announced (p. 82)	AWS Glue Studio provides a visual interface that simplifies the creation of jobs that prepare the data for analysis. The initial version of this guide was published on the same day AWS Glue Studio launched.	September 23, 2020

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.