# Creating Connectors for AWS Marketplace

You can develop your own custom connector software, and then place it on AWS Marketplace to sell to AWS Glue customers.

## Overview of Creating Connectors for AWS Marketplace

AWS Glue Custom Connectors allows you to discover and subscribe to more than 50 connectors on AWS Marketplace. You can also use AWS Glue Spark runtime interfaces to plug-in connectors built for Apache Spark Datasource, Athena federated query, and JDBC APIs. This chapter helps you to build and test custom connectors; and deploy them for connectivity with AWS Glue Spark applications.

1. Create a custom connector as described in Step 1: Developing Marketplace Connectors.
2. Create a new product in AWS Marketplace Step 2: Create your connector product in AWS Marketplace.
3. Package and upload your connector Step 3: Packaging and Uploading Marketplace Connectors.
4. Create a deployment link for your Glue connector product Step 4: Creating a Deployment Link for Marketplace Connectors.
5. Test your connector as described in Step 5: Testing Marketplace Connectors.
6. Validate your connector as described in Step 6: Validating Marketplace Connectors
7. After you've completed the above steps, contact your assigned Technical Account Manager (TAM) to push your product to the public in Step 7: Publish the product to the public.

## Step 1: Developing Marketplace Connectors

You can create connectors for JDBC, Spark, or Athena data sources. Each connector type has different requirements.

- To create a connector for Open Spark data stores, see Developing Spark connectors
- To create a connector for Amazon Athena data stores, see Developing Athena connectors
- To create a connector for JDBC data stores, see Developing JDBC connectors

## Step 2: Create your connector product in AWS Marketplace

Creating a product in AWS Marketplace involves the following steps:

1. Create the product ID.
2. Create the pricing details.
3. For paid products, integrate metering into your product.
4. Add a new version of your product, including:

    1. Add repositories for your containers.
    2. Upload the final containers into the repositories.
    3. Create the first version of the product with your first container images.
5. Update the product information.
6. Publish the product for buyers.

To proceed to packaging and uploading your connector next, you should follow this guide to finish the aforementioned list up until step 4a. By the end of step 4a, you will get an ECR repository url similar to `709825985650.dkr.ecr.us-east-1.amazonaws.com/my-company/salesforce`. Note it down, we will need in the next section.

## Step 3: Packaging and Uploading Marketplace Connectors

This section describes how to create and publish a container product with the required connector jars to AWS Marketplace. The script is stored under the same directory in Github.

1. Setup AWS Command Line Interface (AWS CLI). Refer to Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide* for the instructions.
   You can install either of the following versions:

    - AWS CLI version 2
    - AWS CLI version 1 v1.17.10 or later

1. Install Docker Engine, as described in "Install Docker Engine" in the *Docker Engine* online documentation.
2. Create and start a docker image, as described in "Orientation and setup" in the Docker online documentation.
3. Prepare the config file to provide metadata about the connector that you're publishing. A sample file is provided below. All the keys in the example below are **required**.

```
{"releasetimestamp": "2020-12-21 12:00:00",
 "connectiontype": "MARKETPLACE",
 "classname": "partner.jdbc.salesforce.salesforcedriver",
 "publishername": "partner",
 "connectortype": "JDBC",
 "version": "19.0.7362.0",
 "description": "Partner JDBC Driver for Salesforce",
 "supportinformation": "Please check for this driver's online help."
}
```

4. Download the shell script for creating and publishing a container with the required connector JAR files and config file for the connector JAR files.

   **Tip**
   Copy and run the script in a new folder to avoid conflicts, and for the script to run efficiently.

This script creates a new container with the necessary JAR files and then pushes the container to the newly created Amazon Elastic Container Registry (Amazon ECR) repository.

Here the value of `--image-tag` should be the version your connector jar maps to, you can use other values but it's recommended to use the version to be consistent with the version you're going to create later for the product. **Attention**: The value of `--ecr-repo-name` should be the suffix after `.com/` of the ECR URL we noted in the last step. For example, if the ECR URL is `709825985650.dkr.ecr.us-east-1.amazonaws.com/my-company/salesforce`, then the value should be `my-company/salesforce`.

5. You can run the following command to see the usage examples for the script and how to run the script to publish the docker image to Amazon ECR:

```
bash container_setup.sh --help
```

6. After you've run this script and uploaded the connector to ECR, please note down the final url of the ECR repo in the script output.

## Step 4: Creating a Deployment Link for Marketplace Connectors

At the end of a customer's subscription workflow in the AWS Marketplace, they need a way to activate the connector they just purchased in AWS Glue Studio. This section describes how to prepare a deep link URL, which redirects the user back to the AWS Glue Studio console to activate the connector.

Please follow these steps carefully and reach out to your AWS contact if you have any questions.

1. Gather the information required for the deep link URL. The deep link URL is an absolute URL that points to an endpoint provided by AWS Glue Studio. The base url is:

   https://console.aws.amazon.com/gluestudio/home#/connector/add-connection?*PARAMETERS*
   There are several parameters you need to append to the base URL so that your connector can be integrated with AWS Glue Studio correctly.

   **Connector-related parameters:**

- **connectorName** (required): The name of your connector. You can also include your company name, if you want. The value should be an alphanumeric string that uses a space character as a delimiter. This field is editable by the end-user.

```
connectorName="Virtual Company Simple DB"
```

- **connectorType** (required): The interface type that your connector is built upon. The accepted values are Spark, Athena, or Jdbc. This field is not editable by the end-user.

```
connectorType="Jdbc"
```

- **connectorDescription** (optional): A brief summary about this connector that contains only alphanumeric letters, spaces, and the characters ",", ";", or ".". We recommend limiting this field to 50 words or less. This field is editable by the end-user.

```
connectorDescription="A simple description"
```

- **connectorUrl** (required): The URL for the corresponding Amazon Elastic Container Registry (Amazon ECR) image that contains your connector, namely the ECR repo you published to at Step 3.

```
connectorUrl="https://mp-account-#.dkr.ecr.us-east-
1.amazonaws.com/product_id/container_group_id/myconnectorimage:version_title-latest"
```

- **connectorVersion** (required): The version of your connector. The value should contain only numeric letters delimited by ".". The length limit for this field is 36 characters. This field is not editable by the end-user.

```
connectorVersion="7.5.5"
```

- **connectorClassName** (required): For JDBC connectors, this field should be the class name of your JDBC driver. For Spark connectors, this field should be the format when loading spark data source with the format operator.

```
connectorClassName="some.class.name"
```

For Spark connectors, an example using an alias is:

```
connectorClassName="es"
```

The following is a Spark example using the real marker class name:

```
connectorClassName="net.snowflake.spark.snowflake"
```

For Athena, you would use a class name similar to:

```
connectorClassName="com.amazonaws.athena.connectors.Cloudwatch"
```

**Connection-related parameters:**

- **connectionAccessJdbcURLFormat** (required for JDBC connectors): The JDBC URL templates supported by your connector. If you support more than one template, you can reuse this keyword to specify each of them. Also, each template should start with a name followed by an equal sign. For example, if your connector supports a template that uses a username and password for authentication, then you can put:

```
username_password=jdbc:virtualcomp:simpleDB:user=${Username};password=${Password}
```

The first token, `username_password`, can be any non-empty string. This represents the title presented to the user from a drop-down component in the AWS Glue Studio console. You should make this title easy to read and identifiable. If there is more than one template, make sure this title is unique, for example:

```
username_password_template1= ... username_password_template2= ...
```

For the JDBC URL template, you can insert placeholders for parameter values that user has to fill in at runtime for their specific data store. You should also include a usage guide on your product page to call out these mandatory fields and what values are expected for them. Each placeholder variable should be enclosed in curly braces and prefixed with a dollar sign, for example `${RuntimeKey}`.

If your JDBC URL template contains user name and password parameters, then the placeholder variables for them should *always* be exactly `Username` and `Password`. This is required for AWS Glue Studio to extract them correctly and apply the proper encryption.

A full example looks like this:

```
connectionAccessJdbcURLFormat="username_password=jdbc:virtualcomp:simpleDB:user=${Username};
password=${Password}"
connectionAccessJdbcURLFormat="oauth=jdbc:virtualcomp:simpleDB:OAuthSettingsLocation=${OAuth
SettingsLocation};InitiateOAuth=REFRESH;"
```

- **connectionAccessJdbcURLParamsDelimiter** (required for JDBC connectors): For JDBC connectors, this field specifies the delimiter used to separate parameters in the JDBC URL template. This enables users to add additional JDBC parameter key-value pairs in the AWS Glue Studio console.

  ```
  connectionAccessJdbcURLParamsDelimiter=";"
  ```

1. After you have all the parameters ready, you can join them together with an ampersand (&) and append the result to the base URL. The outcome should look like this (with no spaces between each parameter query):

```
https://console.aws.amazon.com/gluestudio/home#/connector/add-connection
?connectorName="Virtual Company Simple DB"
&connectorType="Jdbc"
&connectorDescription="A virtual company connector that connects to Simple DB"
&connectorUrl="https://mp-account-#.dkr.ecr.us-east-
1.amazonaws.com/product_id/container_group_id/myconnectorimage:version_title-latest"
&connectorVersion="7.5.5"
&connectorClassName="virtualcomp.db.simpledb.driver"
&connectionAccessJdbcURLFormat="username_password=jdbc:virtualcomp:simpleDB:user=${Username
};password=${Password}"
&connectionAccessJdbcURLFormat="oauth=jdbc:virtualcomp:simpleDB:OAuthSettingsLocation=${OAut
hSettingsLocation};InitiateOAuth=REFRESH;"
&connectionAccessJdbcURLParamsDelimiter=";"
```

1. Use an URI encoder such as the one at https://toolbox.googleapps.com/apps/encode_decode/ to encode the parameter values.

This helps to avoid any character escaping issues. Using the example in the previous step, the encoded URL string should look like this (but as a one-line string):

```
https://console.aws.amazon.com/gluestudio/home#/connector/add-connection?connectorNa
me=%22Virtual%20Company%20Simple%20DB%22&connectorType=%22Jdbc%22&connectorDescripti
on=%22A%20vir
tual%20company%20connector%20that%20connects%20to%20Simple%20DB%22&connectorUrl=%22ht
tps://mp-acc
ount-#.dkr.ecr.us-east-1.amazonaws.com/product_id/container_group_id/myconnectorimage:version_title-
latest%22&connectorVersion=%227.5.5%22&connectorClassName=%22virtualcomp.db.simpledb
.driver%22&connectionAccessJdbcURLFormat=%22username_password=jdbc:virtualcomp:simpleDB:use
r=$%7B
Username%7D;password=$%7BPassword%7D%22&connectionAccessJdbcURLFormat=%22oauth=jdbc:
virtualcomp:
simpleDB:OAuthSettingsLocation=$%7BOAuthSettingsLocation%7D;InitiateOAuth=REFRESH;%22&co
nnection
AccessJdbcURLParamsDelimiter=%22%3B%22
```

**Note**
Only the values of the parameters are encoded. You should only encode the values of the parameter (the value to the right of each assignment operator '=' ).

1. Now you have the deep link url, you can resume the product creation. In Step 2 of this guidance, we left at step 4a to create a connector product, now you can continue with step 4b.
2. If you're creating a new version for the first time, at step 4c, you can select to "Add new delivery option".
   In the form prompted, you should be able to see following fields:

   - **Container images**: paste in the ECR URL you've noted from the output of the container-setup.sh script.
   **- Title of delivery option**: Enter Activate in AWS Glue Studio.
   **- Description**: Enter Import jars in AWS Glue jobs..
   **- Usage instructions**: Paste in the sentence below with deep link url you've prepared inserted into the parenthesis.

   ```
   Please subscribe to the product from AWS Marketplace and [Activate the Glue connector from AWS Glue Studio](<paste in your deep url here>)
   ```

   1. Once done, you can submit the version and it will be created.

## Step 5: Testing Marketplace Connectors

Perform the integration tests to test a connector against most AWS Glue features locally before releasing your connector to the AWS Glue ETL connector marketplace.

Refer to the Marketplace Connector Integration Tests Guide on GitHub, which shows you how to:

- Set up the tool
- Configure each test
- Run each test

The results of these tests are using in to provide the validation information for your connector.

## Step 6: Validating Marketplace Connectors

You must validate your connector against AWS Glue supported features in AWS Glue job system before you publish it to AWS Marketplace.

Refer to the Marketplace Connector Validation Guide on GitHub.

1. Complete the steps in the previous section, Step 3: Testing Marketplace Connectors, to perform the validation tests.
2. After you finish the validation testing, complete the steps in the **Reporting** section of the Marketplace Connector Validation Guide.

## Step 7: Publish the product to the public

After all these steps are done, you can finally publish your product so that it will be visible to all AWS customers.