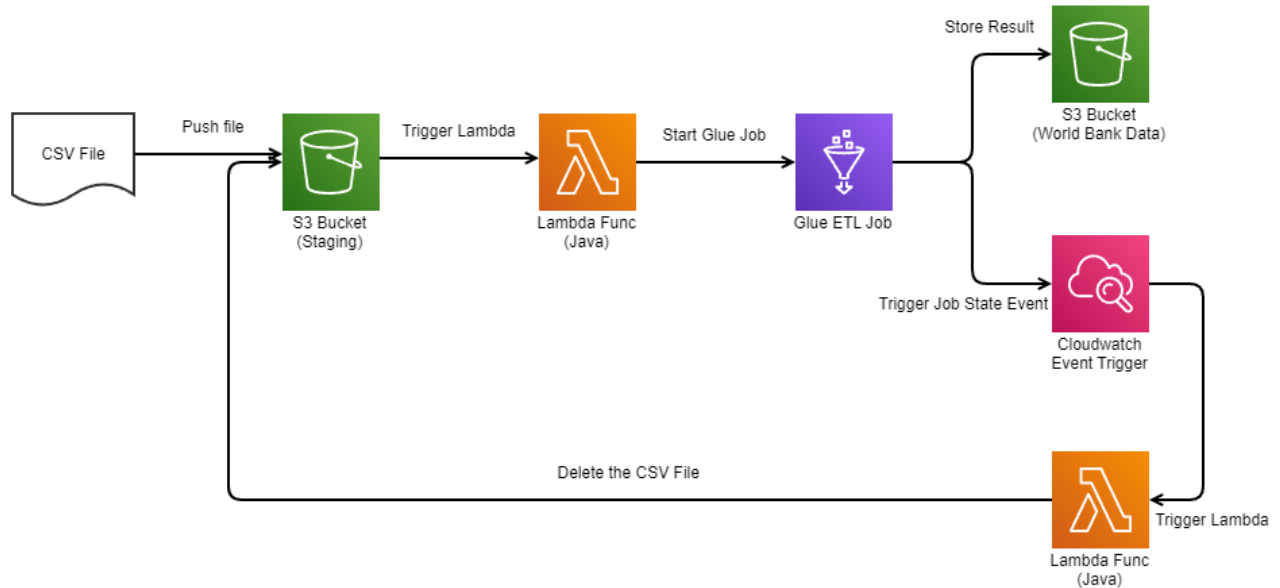


AWS POC 2 - Load Government Debt Total % GDP

Overview



Proof of Concept flow

1. Push the latest .csv to the staging S3 bucket
2. The S3 put object event triggers the Lambda function
3. The Lambda function kicks off the Glue ETL job and exits
4. The Glue ETL runs and stores the transformed data in the World Bank Open Data S3 bucket in Parquet format
5. The job state change event is picked up by CloudWatch and triggers the Lambda function
6. The Lambda function deletes the recently uploaded csv file

S3 Bucket Setup

Bucket overview	
AWS Region Asia Pacific (Sydney) ap-southeast-2	Amazon Resource Name (ARN) arn:aws:s3::poc-staging

Bucket overview	
AWS Region Asia Pacific (Sydney) ap-southeast-2	Amazon Resource Name (ARN) arn:aws:s3::poc-world-bank-data

Staging S3 bucket and the World Bank Data S3 bucket.

Lambda Setup

S3 Bucket Updated Policy

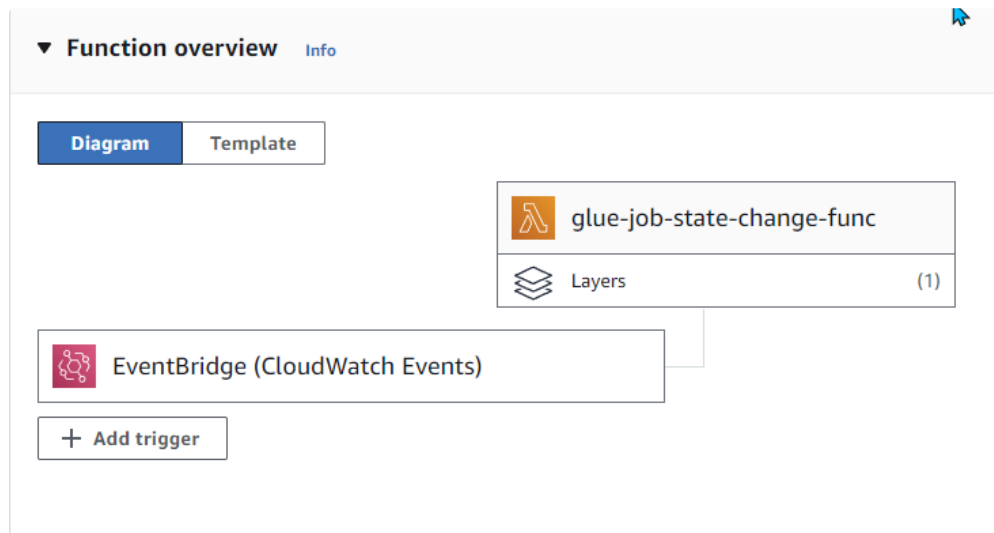
Update the permissions to allow object deletion.

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "logs:PutLogEvents",
8                  "logs:CreateLogGroup",
9                  "logs:CreateLogStream"
10             ],
11             "Resource": "arn:aws:logs:*:*:*"
12         },
13         {
14             "Effect": "Allow",
15             "Action": [
16                 "s3:GetObject",
17                 "s3:PutObject",
18                 "s3:DeleteObject"
19             ],
20             "Resource": "arn:aws:s3:::*/*"
21         }
22     ]
23 }

```

Lambda Function



Instead of a *fat jar* the Lambda function setup has a layer associated with the required JAR files as part of the layer instead.

Layers Info				
Merge order	Name	Layer version	Compatible runtimes	Compatible architectures
1	aws-lambda-sdk-support	1	java21	x86_64

Runtime settings Info		
Runtime Java 21	Handler Info com.mattswart.aws.GlueJobLambdaFuncHandler::handleRequest	Architecture x86_64
▶ Runtime management configuration		

Lambda Function (Java) Deep Dive

Lambda Function Code (Java)

```

1 package com.mattswart.aws;
2
3 import java.util.HashMap;
4
5 import com.amazonaws.services.lambda.runtime.Context;
6 import com.amazonaws.services.lambda.runtime.LambdaLogger;
7 import com.amazonaws.services.lambda.runtime.RequestHandler;
8
9 public class GlueJobLambdaFuncHandler implements RequestHandler<HashMap, String> {
10     @Override
11     public String handleRequest(HashMap event, Context context) {
12         try {
13             LambdaLogger logger = context.getLogger();
14
15             GlueJobStateChangeEvent glueJobEvent = new GlueJobStateChangeEvent();
16             glueJobEvent.parseEventHashMap(event);
17
18             logger.log("Successfully retrieved " + glueJobEvent.toString());
19             logger.log("Job name " + glueJobEvent.getName());
20             logger.log("Job state " + glueJobEvent.getState());
21
22             return "Ok";
23         } catch (Exception e) {
24             throw new RuntimeException(e);
25         }
26     }
27
28 }

```

Glue Job State Change Event (Java)

```

1 package com.mattswart.aws;
2
3 import java.util.HashMap;
4
5 public class GlueJobStateChangeEvent {
6     private String jobName = "";
7     private String severity = "";
8     private String state = "";
9     private String jobRunId = "";
10    private String message = "";
11
12    public boolean parseEventHashMap(HashMap eventHashMap){
13        HashMap detail = (HashMap) eventHashMap.get("detail");

```

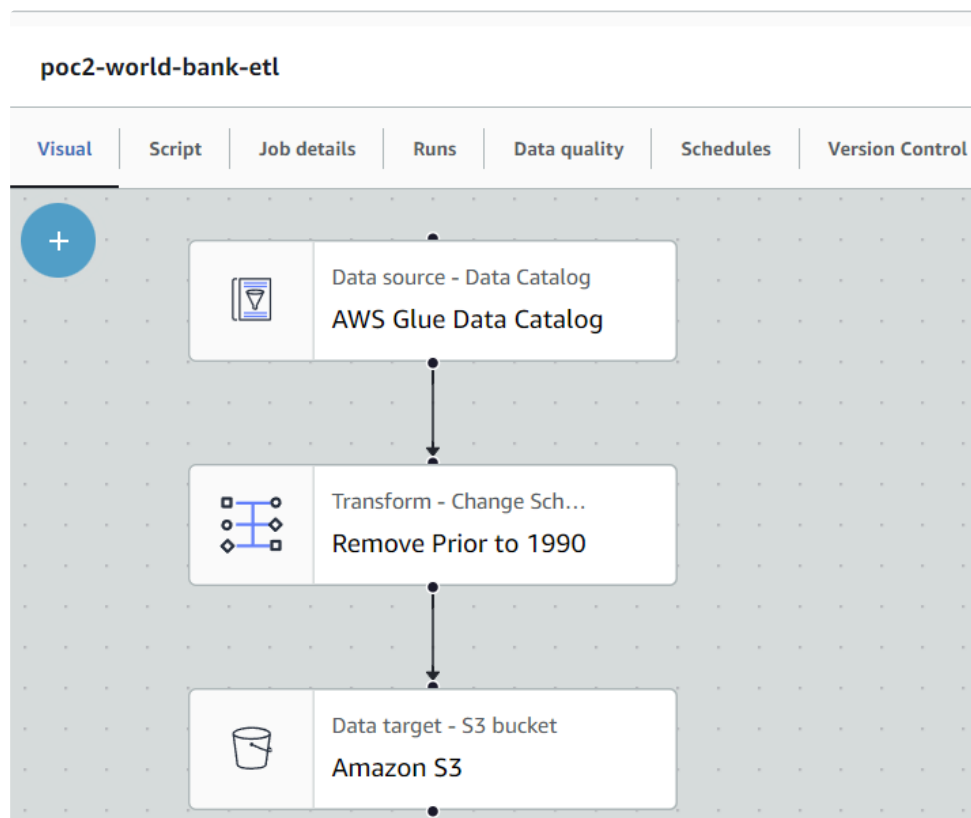
```

14     this.jobName = detail.get("jobName").toString();
15     this.severity = detail.get("severity").toString();
16     this.state = detail.get("state").toString();
17     this.jobRunId = detail.get("jobRunId").toString();
18     this.message = detail.get("message").toString();
19     return true;
20 }
21
22 public String getName(){
23     return this.jobName;
24 }
25
26 public String getSeverity(){
27     return this.severity;
28 }
29 public String getState(){
30     return this.state;
31 }
32 public String getJobRunId(){
33     return this.jobRunId;
34 }
35 public String getMessage(){
36     return this.message;
37 }
38 }

```

Glue Job Deep Dive

Glue Visual ETL



Simple ETL to load the data from the CSV file in the Staging S3 bucket, remove dates prior to 1990 and save the result in the World Bank S3 bucket.

Job Details

Basic properties

Info

Name

poc2-world-bank-etl

Description - optional

Descriptions can be up to 2048 characters long.

IAM Role

Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.

glue-poc-role

Type

The type of ETL job. This is set automatically based on the types of data sources you have selected.

Spark

Glue version

Info

Glue 4.0 - Supports spark 3.3, Scala 2, Python 3

Language

Scala

Worker type

Set the type of predefined worker that is allowed when a job runs.

G 1X
(4vCPU and 16GB RAM)

Automatically scale the number of workers

☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

Requested number of workers

The number of workers you want AWS Glue to allocate to this job.

2

Job Role & Policy

Permissions policies (2)



Info

You can attach up to 10 managed policies.

Q Search

Filter by Type

All types

<input type="checkbox"/> Policy name	Type	Attached entities
<input type="checkbox"/>  AWSGlueServiceRole	AWS managed	1
<input type="checkbox"/>  s3-trigger-aws-poc1	Customer managed	2

The glue-poc-role has policies to allow S3 bucket Read/Write access and also the Amazon managed Glue Service Role.

Database properties

Name

aws-poc2-db

Description

-

Location

-

Tables (1)

View and manage all available tables.

Filter tables

<input type="checkbox"/>	Name	Database	Location	Classification
<input type="checkbox"/>	world-bank-indicators	aws-poc2-db	s3://poc-staging/world-bank-open-data/GC.DOD.TOTL.GD.ZS/	CSV

Table Schema

```
1  [  
2    {  
3      "Name": "country name",  
4      "Type": "string",  
5      "Comment": ""  
6    },  
7    {  
8      "Name": "country code",  
9      "Type": "string",  
10     "Comment": ""  
11   },  
12   {  
13     "Name": "indicator name",  
14     "Type": "string",  
15     "Comment": ""  
16   },  
17   {  
18     "Name": "indicator code",  
19     "Type": "string",  
20     "Comment": ""  
21   },  
22   {  
23     "Name": "1960",  
24     "Type": "decimal",  
25     "Comment": ""  
26   },  
27   {  
28     "Name": "1961",  
29     "Type": "decimal",  
30     "Comment": ""  
31   },  
32   {  
33     "Name": "1962",  
34     "Type": "decimal",  
35     "Comment": ""  
36   },  
37   {  
38     "Name": "1963",  
39     "Type": "decimal",  
40     "Comment": ""  
41   },  
42   {  
43     "Name": "1964",  
44     "Type": "decimal",
```

```
45     "Comment": ""
46 },
47 {
48     "Name": "1965",
49     "Type": "decimal",
50     "Comment": ""
51 },
52 {
53     "Name": "1966",
54     "Type": "decimal",
55     "Comment": ""
56 },
57 {
58     "Name": "1967",
59     "Type": "decimal",
60     "Comment": ""
61 },
62 {
63     "Name": "1968",
64     "Type": "decimal",
65     "Comment": ""
66 },
67 {
68     "Name": "1969",
69     "Type": "decimal",
70     "Comment": ""
71 },
72 {
73     "Name": "1970",
74     "Type": "decimal",
75     "Comment": ""
76 },
77 {
78     "Name": "1971",
79     "Type": "decimal",
80     "Comment": ""
81 },
82 {
83     "Name": "1972",
84     "Type": "decimal",
85     "Comment": ""
86 },
87 {
88     "Name": "1973",
89     "Type": "decimal",
90     "Comment": ""
91 },
92 {
93     "Name": "1974",
94     "Type": "decimal",
95     "Comment": ""
96 },
97 {
98     "Name": "1975",
99     "Type": "decimal",
100     "Comment": ""
101 },
102 {
```



```
103     "Name": "1976",
104     "Type": "decimal",
105     "Comment": ""
106 },
107 {
108     "Name": "1977",
109     "Type": "decimal",
110     "Comment": ""
111 },
112 {
113     "Name": "1978",
114     "Type": "decimal",
115     "Comment": ""
116 },
117 {
118     "Name": "1979",
119     "Type": "decimal",
120     "Comment": ""
121 },
122 {
123     "Name": "1980",
124     "Type": "decimal",
125     "Comment": ""
126 },
127 {
128     "Name": "1981",
129     "Type": "decimal",
130     "Comment": ""
131 },
132 {
133     "Name": "1982",
134     "Type": "decimal",
135     "Comment": ""
136 },
137 {
138     "Name": "1983",
139     "Type": "decimal",
140     "Comment": ""
141 },
142 {
143     "Name": "1984",
144     "Type": "decimal",
145     "Comment": ""
146 },
147 {
148     "Name": "1985",
149     "Type": "decimal",
150     "Comment": ""
151 },
152 {
153     "Name": "1986",
154     "Type": "decimal",
155     "Comment": ""
156 },
157 {
158     "Name": "1987",
159     "Type": "decimal",
160     "Comment": ""
```

```
161 },
162 {
163     "Name": "1988",
164     "Type": "decimal",
165     "Comment": ""
166 },
167 {
168     "Name": "1989",
169     "Type": "decimal",
170     "Comment": ""
171 },
172 {
173     "Name": "1990",
174     "Type": "decimal",
175     "Comment": ""
176 },
177 {
178     "Name": "1991",
179     "Type": "decimal",
180     "Comment": ""
181 },
182 {
183     "Name": "1992",
184     "Type": "decimal",
185     "Comment": ""
186 },
187 {
188     "Name": "1993",
189     "Type": "decimal",
190     "Comment": ""
191 },
192 {
193     "Name": "1994",
194     "Type": "decimal",
195     "Comment": ""
196 },
197 {
198     "Name": "1995",
199     "Type": "decimal",
200     "Comment": ""
201 },
202 {
203     "Name": "1996",
204     "Type": "decimal",
205     "Comment": ""
206 },
207 {
208     "Name": "1997",
209     "Type": "decimal",
210     "Comment": ""
211 },
212 {
213     "Name": "1998",
214     "Type": "decimal",
215     "Comment": ""
216 },
217 {
218     "Name": "1999",
```

```
219     "Type": "decimal",
220     "Comment": ""
221 },
222 {
223     "Name": "2000",
224     "Type": "decimal",
225     "Comment": ""
226 },
227 {
228     "Name": "2001",
229     "Type": "decimal",
230     "Comment": ""
231 },
232 {
233     "Name": "2002",
234     "Type": "decimal",
235     "Comment": ""
236 },
237 {
238     "Name": "2003",
239     "Type": "decimal",
240     "Comment": ""
241 },
242 {
243     "Name": "2004",
244     "Type": "decimal",
245     "Comment": ""
246 },
247 {
248     "Name": "2005",
249     "Type": "decimal",
250     "Comment": ""
251 },
252 {
253     "Name": "2006",
254     "Type": "decimal",
255     "Comment": ""
256 },
257 {
258     "Name": "2007",
259     "Type": "decimal",
260     "Comment": ""
261 },
262 {
263     "Name": "2008",
264     "Type": "decimal",
265     "Comment": ""
266 },
267 {
268     "Name": "2009",
269     "Type": "decimal",
270     "Comment": ""
271 },
272 {
273     "Name": "2010",
274     "Type": "decimal",
275     "Comment": ""
276 },
```

```
277 {
278     "Name": "2011",
279     "Type": "decimal",
280     "Comment": ""
281 },
282 {
283     "Name": "2012",
284     "Type": "decimal",
285     "Comment": ""
286 },
287 {
288     "Name": "2013",
289     "Type": "decimal",
290     "Comment": ""
291 },
292 {
293     "Name": "2014",
294     "Type": "decimal",
295     "Comment": ""
296 },
297 {
298     "Name": "2015",
299     "Type": "decimal",
300     "Comment": ""
301 },
302 {
303     "Name": "2016",
304     "Type": "decimal",
305     "Comment": ""
306 },
307 {
308     "Name": "2017",
309     "Type": "decimal",
310     "Comment": ""
311 },
312 {
313     "Name": "2018",
314     "Type": "decimal",
315     "Comment": ""
316 },
317 {
318     "Name": "2019",
319     "Type": "decimal",
320     "Comment": ""
321 },
322 {
323     "Name": "2020",
324     "Type": "decimal",
325     "Comment": ""
326 },
327 {
328     "Name": "2021",
329     "Type": "decimal",
330     "Comment": ""
331 },
332 {
333     "Name": "2022",
334     "Type": "decimal",
```

```

335     "Comment": ""
336   }
337 ]

```




Glue Job Code (Scala)

```

1  import com.amazonaws.services.glue.GlueContext
2  import com.amazonaws.services.glue.MappingSpec
3  import com.amazonaws.services.glue.errors.CallSite
4  import com.amazonaws.services.glue.util.GlueArgParser
5  import com.amazonaws.services.glue.util.Job
6  import com.amazonaws.services.glue.util.JsonOptions
7  import org.apache.spark.SparkContext
8  import scala.collection.JavaConverters._
9
10 object GlueApp {
11   def main(sysArgs: Array[String]) {
12     val spark: SparkContext = new SparkContext()
13     val glueContext: GlueContext = new GlueContext(spark)
14     // @params: [JOB_NAME]
15     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
16     Job.init(args("JOB_NAME"), glueContext, args.asJava)
17     // Script generated for node AWS Glue Data Catalog
18     val AWSGlueDataCatalog_node1708237359808 = glueContext.getCatalogSource(database="aws-poc2-db", tableName="w
19
20     // Script generated for node Remove Prior to 1990
21     val RemovePriorTo1990_node1708238527332 = AWSGlueDataCatalog_node1708237359808.applyMapping(mappings=Seq(("1
22
23     // Script generated for node Amazon S3
24     val AmazonS3_node1708238624148 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions(""{
25
26     Job.commit()
27   }
28 }

```

CloudWatch Glue Job State Change Rule

Rule details info			
Rule name test-glue-job-finished	Status  Enabled	Event bus name default	Type Standard
Description	Rule ARN  arn:aws:events:ap-southeast-2:138425731864:rule/test-glue-job-finished	Event bus ARN  arn:aws:events:ap-southeast-2:138425731864:event-bus/default	

Event Pattern

```

1  {
2    "source": ["aws.glue"],
3    "detail-type": ["Glue Job State Change"]
4  }

```

--

IdWatch Events): [test-glue-job-f](#)
east-2:138425731864:rule/test-glue-job-f
)

--

-
"State Change"

-finished

ents.amazonaws.com

vents_test-glue-job-finished_Id41d

rules/test-glue-job-finished

Resources

Proof of Concept Github	https://github.com/marthinusswart/aws-proof-of-concepts/tree/main
World Bank Open Data - Central Government Debt Total	https://data.worldbank.org/indicator/GC.DOD.TOTL.GD.ZS?view=chart