

Event-Driven Data Validation and Processing Pipeline Using S3, Step Functions, and ECS



Objective

- Trigger: Use Amazon EventBridge to monitor S3 for newly created objects and trigger an AWS Step Function.
- Validate: Perform data validation using Python scripts with Pandas, ensuring data integrity.
- Move: Transfer validated data to a designated S3 location.
- Condition Check: Evaluate the output of the first validation job to determine readiness for further processing.
- Process: Trigger the second task to perform additional data transformations using Python.
- Run Tasks: Execute both Python scripts as containerized tasks on AWS Fargate with images stored in Amazon ECR.
- Track: Record the status and timestamps of the pipeline's execution in Amazon DynamoDB.

Tools Used

- Amazon S3: As the source and destination for raw and validated data.
- Amazon EventBridge: For detecting object creation events in S3 and triggering the pipeline.
- AWS Step Functions: For orchestrating the pipeline's execution, including validation and condition checks.
- Python: For writing data validation and processing logic using Pandas.
- Amazon ECS (Fargate): For running Python scripts in a serverless, containerized environment.
- Amazon ECR: For storing container images used in ECS tasks.
- Amazon DynamoDB: For logging pipeline status and tracking execution timestamps.
- Pandas: For performing data validation and transformation operations.
- CloudWatch: For monitoring the execution flow and capturing logs.



AWS Cloud



PC



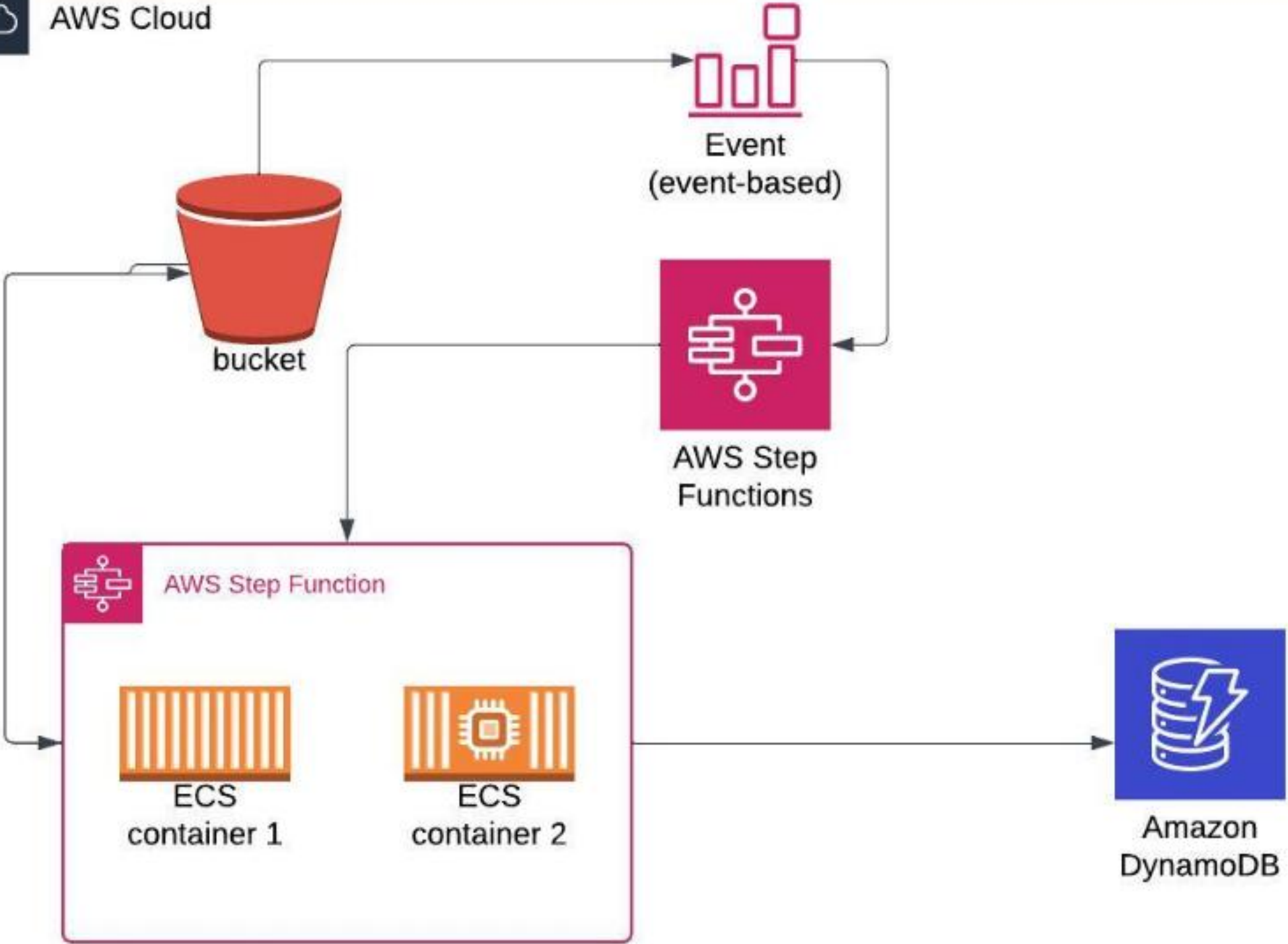
ECR
registry



Amazon ECS
Anywhere



AWS Cloud





- Learn
- Sandbox
- Quick starts
- ▼ **Buses**
 - Event buses
 - Rules**
 - Global endpoints
 - Archives
 - Replays
- ▼ **Pipes**
 - Pipes
- ▼ **Scheduler**
 - Schedules
 - Schedule groups
- ▼ **Integration**
 - Partner event sources
 - API destinations
 - Connections **Updated**

Rules

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

Select event bus

Event bus

Select or enter event bus name

default ▼

Rules (1)



Delete

Enable

Edit

CloudFormation Template ▼

Create rule

Find rules

Any status ▼

< 1 > ⚙️

<input type="checkbox"/>	Name ▲	Status ▼	Type ▼	ARN ▼	Description ▼
<input type="checkbox"/>	trigger-ecs-tasks	Enabled	Standard	arn:aws:events:us-east-1:418272783111:rule/trigger-ecs-tasks	xzzz



Learn
Sandbox
Quick starts

▼ Buses

Event buses

Rules

Global endpoints

Archives

Replays

▼ Pipes

Pipes

▼ Scheduler

Schedules

Schedule groups

▼ Integration

Partner event sources

API destinations

Connections **Updated**

Event pattern

Targets

Monitoring

Tags

Event pattern [Info](#)

Edit

```
1 {  
2   "source": ["aws.s3"],  
3   "detail-type": ["Object Created"],  
4   "detail": {  
5     "bucket": {  
6       "name": ["gd-aws-de-labs"]  
7     },  
8     "object": {  
9       "key": [{  
10        "prefix": "ecommerce-data/new/order_items"  
11      }]  
12    }  
13  }  
14 }
```

Copy



Objects

Properties

Objects (3) Info

Copy S3 URI

Copy URL

Download

Open

Delete

Actions




Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 order_items_20240601.csv	csv	December 27, 2024, 14:20:28 (UTC+05:30)	168.8 KB	Standard
<input type="checkbox"/>	 orders_20240601.csv	csv	December 27, 2024, 13:58:12 (UTC+05:30)	94.5 KB	Standard
<input type="checkbox"/>	 products.csv	csv	December 27, 2024, 13:58:23 (UTC+05:30)	4.1 MB	Standard



Step Functions

State machines

Activities

Developer resources

Online learning workshop

Local Development

Data flow simulator

Feature spotlight

Documentation

Join our feedback panel

Arn

arn:aws:states:us-east-1:418272783111:stateMachine:testing

IAM role ARN

arn:aws:iam::418272783111:role/step-functiontesting

Type

Standard

Status

Active

Creation date

Dec 27, 2024, 13:37:56 (UTC+05:30)

X-Ray tracing

Disabled

Executions

Monitoring

Logging

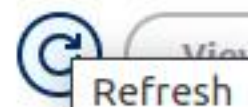
Definition

Aliases

Versions

Tags

Executions (0/2)



Refresh

View details

Stop execution

Redrive

Start execution

Filter executions by property

Filter by status

Last 15 months

Local timezone

2 matches

< 1 >



<input type="checkbox"/>	Name	Status	Start Time (local)	End Time (local)	Dura
<input type="checkbox"/>	e371b047-2ca5-9879-b99d-73133e1fe1...	Running	Dec 27, 2024, 14:20:28	-	-
<input type="checkbox"/>	ea4f1378-aa24-4e5e-a78a-7b80eed7bb9a	Succeeded	Dec 27, 2024, 13:38:13	Dec 27, 2024, 13:39:55	C





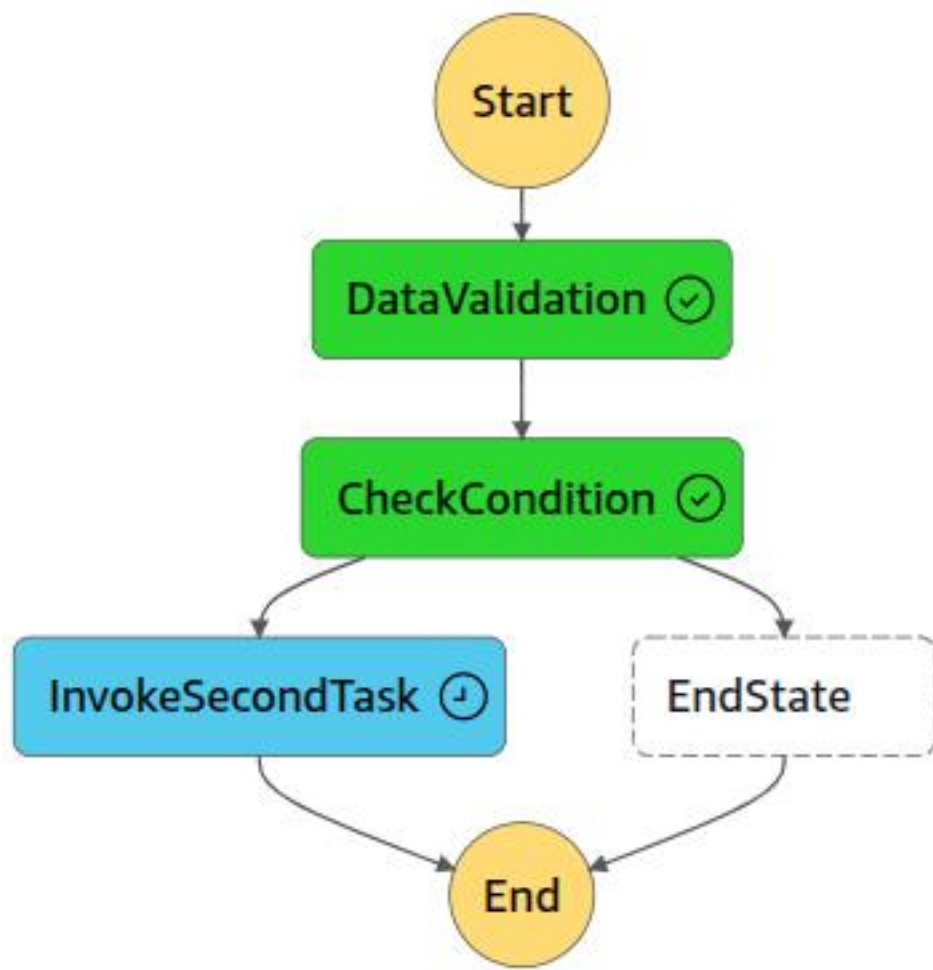
Step Functions

[State machines](#)[Activities](#)

Developer resources

[Online learning workshop](#)[Local Development](#)[Data flow simulator](#)[Feature spotlight](#)[Documentation](#)[Join our feedback panel](#)

Graph view

[Actions](#)

In progress



Failed



Caught error



Canceled



Succeeded

Step details

Choose a step to view its details.

[Event view](#)[State view](#)

step-functions > {} step-function.json > {} States > {} DataValidation > {} Parameters > {} Overrides > [] ContainerOverrides > {} 0

```
1  {
2    "Comment": "A Step Function to process ECS tasks and handle output conditions",
3    "StartAt": "DataValidation",
4    "States": {
5      "DataValidation": {
6        "Type": "Task",
7        "Resource": "arn:aws:states:::ecs:runTask.sync",
8        "Parameters": {
9          "Cluster": "ecommerce-data-ingestion-cluster",
10         "TaskDefinition": "arn:aws:ecs:us-east-1:418272783111:task-definition/task-data-validation:4",
11         "LaunchType": "FARGATE",
12         "Overrides": {
13           "ContainerOverrides": [
14             {
15               "Name": "container-data-validation",
16               "Environment": [
17                 {
18                   "Name": "TASK_TOKEN",
19                   "Value.$": "$$.Task.Token"
20                 }
21               ]
22             }
23           ]
24         },
25         "NetworkConfiguration": {
26           "AwsvpcConfiguration": {
27             "Subnets": [
28               "subnet-09fd1652b231f5ca0",
29               "subnet-0cdf12a5d9170a6ff"
30             ],
31             "AssignPublicIp": "ENABLED"
32           }
33         }
34       }
35     }
36   }
```



step-functions > {} step-function.json > {} States > {} DataValidation > {} Parameters > {} Overrides > [] ContainerOverrides > {} 0

```
4   "States": {
5     "DataValidation": {
8       "Parameters": {
12        "Overrides": {
23          },
24      },
25      "NetworkConfiguration": {
26        "AwsVpcConfiguration": {
27          "Subnets": [
28            "subnet-09fd1652b231f5ca0",
29            "subnet-0cdf12a5d9170a6ff"
30          ],
31          "AssignPublicIp": "ENABLED"
32        }
33      },
34    },
35    "Next": "CheckCondition",
36    "ResultPath": "$.taskOutput"
37  },
38  "CheckCondition": {
39    "Type": "Choice",
40    "Choices": [
41      {
42        "And": [
43          {
44            "Variable": "$.taskOutput.orders",
45            "NumericEquals": 1
46          },
47          {
48            "Variable": "$.taskOutput.order_items",
49            "NumericEquals": 1
50          },
51        ]
52      }
53    ]
54  }
55 }
```



step-functions > {} step-function.json > {} States > {} DataValidation > {} Parameters > {} Overrides > [] ContainerOverrides > {} 0

```
4    "States": {
38      "CheckCondition": {
40        "Choices": [
41          {
42            "And": [
43              {
44                "Variable": "$.taskOutput.orders",
45                "NumericEquals": 1
46              },
47              {
48                "Variable": "$.taskOutput.order_items",
49                "NumericEquals": 1
50              },
51              {
52                "Variable": "$.taskOutput.products",
53                "NumericEquals": 1
54              }
55            ],
56            "Next": "InvokeSecondTask"
57          }
58        ],
59        "Default": "EndState"
60      },
61      "InvokeSecondTask": {
62        "Type": "Task",
63        "Resource": "arn:aws:states:::ecs:runTask.sync",
64        "Parameters": {
65          "Cluster": "ecommerce-data-ingestion-cluster",
66          "TaskDefinition": "arn:aws:ecs:us-east-1:418272783111:task-definition/task-etl-calculations:3",
67          "LaunchType": "FARGATE",
68          "NetworkConfiguration": {
```

step-functions > {} step-function.json > {} States > {} DataValidation > {} Parameters > {} Overrides > [] ContainerOverrides > {} 0

```
4   "States": {
38     "CheckCondition": {
58       },
59     "Default": "EndState"
60   },
61   "InvokeSecondTask": {
62     "Type": "Task",
63     "Resource": "arn:aws:states:::ecs:runTask.sync",
64     "Parameters": {
65       "Cluster": "ecommerce-data-ingestion-cluster",
66       "TaskDefinition": "arn:aws:ecs:us-east-1:418272783111:task-definition/task-etl-calculations:3",
67       "LaunchType": "FARGATE",
68       "NetworkConfiguration": {
69         "AwsvpcConfiguration": {
70           "Subnets": [
71             "subnet-09fd1652b231f5ca0",
72             "subnet-0cdf12a5d9170a6ff"
73           ],
74           "AssignPublicIp": "ENABLED"
75         }
76       }
77     },
78     "End": true
79   },
80   "EndState": {
81     "Type": "Pass",
82     "End": true
83   }
84 }
85 }
```




```
1  import boto3
2  import pandas as pd
3  import logging
4  import json
5  import os
6
7  logging.basicConfig(level=logging.INFO)
8  logger = logging.getLogger()
9
10 s3_bucket = 'gd-aws-de-labs'
11
12 folders = {
13     'orders': 'ecommerce-data/new/orders',
14     'order_items': 'ecommerce-data/new/order_items',
15     'products': 'ecommerce-data/new/products'
16 }
17
18 ready_folders = {
19     'orders': 'ecommerce-data/orders',
20     'order_items': 'ecommerce-data/order_items',
21     'products': 'ecommerce-data/products'
22 }
23
24 def list_files(bucket, prefix):
25     s3_client = boto3.client('s3')
26     paginator = s3_client.get_paginator('list_objects_v2')
27     response_iterator = paginator.paginate(Bucket=bucket, Prefix=prefix)
28
29     files = []
30     for page in response_iterator:
31         if 'Contents' in page:
32             files += [content['Key'] for content in page['Contents'] if content['Key'].endswith('.csv')]
```



```
24 def list_files(bucket, prefix):
25
26     paginator = s3_client.get_paginator('list_objects_v2')
27     response_iterator = paginator.paginate(Bucket=bucket, Prefix=prefix)
28
29     files = []
30     for page in response_iterator:
31         if 'Contents' in page:
32             files += [content['Key'] for content in page['Contents'] if content['Key'].endswith('.csv')]
33     return files
34
35 def read_from_s3(bucket, file_key):
36     s3_client = boto3.client('s3')
37     obj = s3_client.get_object(Bucket=bucket, Key=file_key)
38     df = pd.read_csv(obj['Body'])
39     if 'created_at' in df.columns:
40         df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce', utc=True)
41         df['created_at'] = df['created_at'].dt.tz_localize(None)
42     return df
43
44 def check_format(df, column_formats, file_key):
45     missing_columns = [col for col in column_formats if col not in df.columns]
46     if missing_columns:
47         missing_columns_str = ", ".join(missing_columns)
48         logger.error(f"Missing columns: {missing_columns_str} in {file_key} dataset")
49         return False
50     for column, expected_type in column_formats.items():
51         if column in df.columns:
52             if not pd.api.types.is_dtype_equal(df[column].dtype, expected_type):
53                 logger.error(f"Format check failed for column {column}: expected {expected_type}, found {df[column].dtype}")
54                 return False
55         else:
```



```
43
44 def check_format(df, column_formats, file_key):
45     missing_columns = [col for col in column_formats if col not in df.columns]
46     if missing_columns:
47         missing_columns_str = ", ".join(missing_columns)
48         logger.error(f"Missing columns: {missing_columns_str} in {file_key} dataset")
49         return False
50     for column, expected_type in column_formats.items():
51         if column in df.columns:
52             if not pd.api.types.is_dtype_equal(df[column].dtype, expected_type):
53                 logger.error(f"Format check failed for column {column}: expected {expected_type}, found {df[column].dtype}")
54                 return False
55         else:
56             logger.error(f"Column {column} is missing from the DataFrame in {file_key} dataset.")
57             return False
58     return True
59
60 def move_file(s3_client, bucket, file_key, ready_prefix):
61     copy_source = {'Bucket': bucket, 'Key': file_key}
62     new_key = ready_prefix + '/' + file_key.split('/')[-1]
63     s3_client.copy_object(CopySource=copy_source, Bucket=bucket, Key=new_key)
64     s3_client.delete_object(Bucket=bucket, Key=file_key)
65
66 def main():
67     task_token = os.getenv('TASK_TOKEN')
68     if not task_token:
69         logger.error("TASK_TOKEN environment variable is missing")
70         return
71
72     results = {"orders": 0, "order_items": 0, "products": 0}
73     s3_client = boto3.client('s3')
```

```
66 def main():
67
68     if not task_token:
69         logger.error("TASK_TOKEN environment variable is missing")
70         return
71
72     results = {"orders": 0, "order_items": 0, "products": 0}
73     s3_client = boto3.client('s3')
74
75     column_formats = {
76         'orders': {
77             'order_id': 'int64',
78             'user_id': 'int64',
79             'status': 'object',
80             'created_at': 'datetime64[ns]',
81             'num_of_item': 'int64'
82         },
83         'order_items': {
84             'id': 'int64',
85             'order_id': 'int64',
86             'user_id': 'int64',
87             'product_id': 'int64',
88             'created_at': 'datetime64[ns]'
89         },
90         'products': {
91             'id': 'int64',
92             'sku': 'object',
93             'cost': 'float64',
94             'category': 'object',
95             'name': 'object',
96             'retail_price': 'float64',
97             'department': 'object'
```

```
def main():
    """
    Main function to process data from S3 and store it in a database.
    """
    # Get task token
    task_token = os.getenv('TASK_TOKEN')
    if not task_token:
        logger.error("TASK_TOKEN environment variable is missing")
        return

    # Initialize S3 client
    s3_client = boto3.client('s3')

    # Initialize database client
    db_client = DatabaseClient()

    # Get data from S3
    data = get_data_from_s3(s3_client)

    # Process data
    process_data(data, db_client)

    # Log results
    log_results(data)
```



```
66 def main():
92     'sku': 'object',
93     'cost': 'float64',
94     'category': 'object',
95     'name': 'object',
96     'retail_price': 'float64',
97     'department': 'object'
98 }
99 }
100
101 for folder_key in folders.keys():
102     prefix = folders[folder_key]
103     files = list_files(s3_bucket, prefix)
104
105     for file_name in files:
106         df = read_from_s3(s3_bucket, file_name)
107         format_result = check_format(df, column_formats[folder_key], folder_key)
108         if format_result:
109             results[folder_key] = (variable) s3_client: _ | Any
110             move_file(s3_client, s3_bucket, file_name, ready_folders[folder_key])
111
112 logger.info(f"Results: {json.dumps(results)}")
113
114 stepfunctions_client = boto3.client('stepfunctions')
115 stepfunctions_client.send_task_success({
116     taskToken=task_token,
117     output=json.dumps(results)
118 })
119 logger.info("Task output sent to Step Functions successfully")
120
121 if __name__ == '__main__':
```


- ✓ ECS-STEP-PYTHON
 - > data
 - ✓ docker-data-validity
 - 📄 app.py
 - 📄 Dockerfile**
 - ☰ requirements.txt
 - ✓ docker-dynamo-etl-wrangler
 - 📄 app.py
 - 📄 Dockerfile
 - ☰ requirements.txt
 - ✓ step-functions
 - { } step-function.json
 - { } step-functions-iam-execution-pol...
 - \$ docker-commands.sh
 - { } event-pattern.json
 - 📄 test.ipynb

> OUTLINE

> TIMELINE

docker-data-validity > Dockerfile

```
1 FROM python:3.10
2
3 WORKDIR /usr/src/app
4
5 COPY . .
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 CMD ["python3", "app.py"]
```

✓ ECS-STEP-PYTHON

> data

✓ docker-data-validity

app.py

Dockerfile

requirements.txt

✓ docker-dynamo-etl-wrangler

app.py

Dockerfile

requirements.txt

✓ step-functions

step-function.json

step-functions-iam-execution-pol...

\$ docker-commands.sh

event-pattern.json

test.ipynb

> OUTLINE

> TIMELINE

docker-data-validity > requirements.txt

1 pandas

2 boto3

Ln 2, Col 6 - Screen 4 - UTF-8 - 15 - pip requirements

docker-dynamo-etl-wrangler > app.py > ...

```
1  import pandas as pd
2  import awswrangler as wr
3  import boto3
4  from decimal import Decimal, Inexact, Rounded
5  import logging
6
7  logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
8  logger = logging.getLogger()
9
10 base_path = "s3://gd-aws-de-labs/ecommerce-data/"
11 orders_path = f"{base_path}orders/"
12 order_items_path = f"{base_path}order_items/"
13 products_path = f"{base_path}products/"
14
15 orders_archive_path = f"{base_path}archived/"
16 order_items_archive_path = f"{base_path}archived/"
17 products_archive_path = f"{base_path}archived/"
18
19 try:
20     logger.info("Reading datasets from S3")
21     orders = wr.s3.read_csv(path=orders_path)
22     order_items = wr.s3.read_csv(path=order_items_path)
23     products = wr.s3.read_csv(path=products_path)
24 except Exception as e:
25     logger.error(f"Error reading datasets: {e}")
26     raise
27 try:
28     logger.info("Joining datasets")
29     df = orders.merge(order_items, on='order_id', how='inner', suffixes=('_order', '_item'))
30     df = df.merge(products, left_on="product_id", right_on="id", how='inner', suffixes=('', '_product'))
31
32     df.rename(columns={
        'created_at_order': 'order_date',
        'status_order': 'order_status',
        'id_product': 'product_id',
        'name_product': 'product_name',
        'price_product': 'product_price',
        'created_at_item': 'item_date',
        'status_item': 'item_status',
        'id_order': 'order_id',
        'id_item': 'item_id'
    }, inplace=True)
```



```
27 try:
28     logger.info("Joining datasets")
29     df = orders.merge(order_items, on='order_id', how='inner', suffixes=('_order', '_item'))
30     df = df.merge(products, left_on="product_id", right_on="id", how='inner', suffixes=('', '_product'))
31
32     df.rename(columns={'created_at_order': 'order_date', 'status_order': 'order_status'}, inplace=True)
33
34     df['order_date'] = pd.to_datetime(df['order_date'].str.replace(' UTC', ''), format='%Y-%m-%d %H:%M:%S', errors='coerce')
35
36     grouped = df.groupby([df['order_date'].dt.date, 'category'])
37
38 except Exception as e:
39     logging.error(f"Error processing data: {e}")
40     raise
41
42 def safe_convert_decimal(val):
43     try:
44         return Decimal(str(val))
45     except:
46         return Decimal(str(round(val, 2)))
47
48 def calculate_kpis(group):
49     total_revenue = (group['sale_price'] * group['num_of_item']).sum()
50     avg_order_value = group['sale_price'].mean()
51     avg_return_rate = group[group['order_status'] == 'Returned']['order_id'].nunique() / group['order_id'].nunique()
52     return pd.Series({
53         'daily_revenue': safe_convert_decimal(total_revenue),
54         'avg_order_value': safe_convert_decimal(avg_order_value),
55         'avg_return_rate': safe_convert_decimal(avg_return_rate)
56     })
57
58 try:
```



```
46         return Decimal(str(round(val, 2)))
47
48 def calculate_kpis(group):
49     total_revenue = (group['sale_price'] * group['num_of_item']).sum()
50     avg_order_value = group['sale_price'].mean()
51     avg_return_rate = group[group['order_status'] == 'Returned']['order_id'].nunique() / group['order_id'].nunique()
52     return pd.Series({
53         'daily_revenue': safe_convert_decimal(total_revenue),
54         'avg_order_value': safe_convert_decimal(avg_order_value),
55         'avg_return_rate': safe_convert_decimal(avg_return_rate)
56     })
57
58 try:
59     df_category_wise_summary = grouped.apply(calculate_kpis).reset_index()
60 except Exception as e:
61     logger.error(f"Error calculating category wise KPIS: {e}")
62     raise
63
64 def calculate_order_kpis(group):
65     total_orders = group['order_id'].nunique()
66     total_revenue = (group['sale_price'] * group['num_of_item']).sum()
67     total_items_sold = group['num_of_item'].sum()
68     return_rate = group[group['order_status'] == 'Returned']['order_id'].nunique() / total_orders
69     unique_customers = group['user_id_order'].nunique()
70
71     return pd.Series({
72         'total_orders': total_orders,
73         'total_revenue': safe_convert_decimal(total_revenue),
74         'total_items_sold': total_items_sold,
75         'return_rate': safe_convert_decimal(return_rate),
76         'unique_customers': unique_customers
```



```
64 def calculate_order_kpis(group):
65
66     return pd.Series({
67         'total_orders': total_orders,
68         'total_revenue': safe_convert_decimal(total_revenue),
69         'total_items_sold': total_items_sold,
70         'return_rate': safe_convert_decimal(return_rate),
71         'unique_customers': unique_customers
72     })
73
74
75
76
77
78
79 try:
80     df_daily_order_summary = df.groupby(df['order_date'].dt.date).apply(calculate_order_kpis).reset_index()
81 except Exception as e:
82     logger.error(f"Error calculating daily order KPIs: {e}")
83     raise
84
85
86 dynamodb = boto3.resource('dynamodb')
87 kpi_table = dynamodb.Table('category_wise_summary')
88 order_kpi_table = dynamodb.Table('daily_order_summary')
89
90
91
92
93
94
95 def upsert_dynamodb(table, item):
96     try:
97         table.put_item(Item=item)
98     except Exception as e:
99         logging.error(f"Error upserting data into DynamoDB: {e}")
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```


docker-dynamo-etl-wrangler > app.py > ...

```
93     logging.error(f"Error upserting data into DynamoDB: {e}")
94
95     try:
96         logger.info("Inserting category-wise KPI data into DynamoDB")
97         for index, row in df_category_wise_summary.iterrows():
98             kpi_data = {
99                 'category': row['category'],
100                 'order_date': str(row['order_date']),
101                 'daily_revenue': safe_convert_decimal(row['daily_revenue']),
102                 'avg_order_value': safe_convert_decimal(row['avg_order_value']),
103                 'avg_return_rate': safe_convert_decimal(row['avg_return_rate'])
104             }
105             upsert_dynamodb(kpi_table, kpi_data)
106     except Exception as e:
107         logger.error(f"Error inserting category-wise KPI data: {e}")
108         raise
109
110     try:
111         logger.info("Inserting daily order KPI data into DynamoDB")
112         for index, row in df_daily_order_summary.iterrows():
113             order_kpi_data = {
114                 'order_date': str(row['order_date']),
115                 'total_orders': row['total_orders'],
116                 'total_revenue': safe_convert_decimal(row['total_revenue']),
117                 'total_items_sold': row['total_items_sold'],
118                 'return_rate': safe_convert_decimal(row['return_rate']),
119                 'unique_customers': row['unique_customers']
120             }
121             upsert_dynamodb(order_kpi_table, order_kpi_data)
122     except Exception as e:
123         logger.error(f"Error inserting daily order KPI data: {e}")
124         raise
```


docker-dynamo-etl-wrangler > app.py > ...

```
119         unique_customers : row[ unique_customers ]
120     }
121     upsert_dynamodb(order_kpi_table, order_kpi_data)
122 except Exception as e:
123     logger.error(f"Error inserting daily order KPI data: {e}")
124     raise
125
126 logger.info("KPIs calculated and saved successfully to DynamoDB tables category_wise_summary and daily_order_summary")
127
128 s3_client = boto3.client('s3')
129
130 def move_s3_files(source_path, archive_path):
131     s3_client = boto3.client('s3')
132     bucket = 'gd-aws-de-labs'
133     source_prefix = source_path.replace("s3://gd-aws-de-labs/", "")
134     archive_prefix = archive_path.replace("s3://gd-aws-de-labs/", "")
135
136     try:
137         response = s3_client.list_objects_v2(Bucket=bucket, Prefix=source_prefix)
138         if 'Contents' in response:
139             for obj in response['Contents']:
140                 source_key = obj['Key']
141                 archive_key = source_key.replace(source_prefix, archive_prefix)
142                 if not archive_key.startswith(archive_prefix):
143                     archive_key = archive_prefix + source_key.split('/')[-1]
144
145                 s3_client.copy_object(
146                     CopySource={'Bucket': bucket, 'Key': source_key},
147                     Bucket=bucket,
148                     Key=archive_key
149                 )
150                 s3_client.delete_object(Bucket=bucket, Key=source_key)
```



```

130 def move_s3_files(source_path, archive_path):
131     s3_client = boto3.client('s3')
132     bucket = 'gd-aws-de-labs'
133     source_prefix = source_path.replace("s3://gd-aws-de-labs/", "")
134     archive_prefix = archive_path.replace("s3://gd-aws-de-labs/", "")
135
136     try:
137         response = s3_client.list_objects_v2(Bucket=bucket, Prefix=source_prefix)
138         if 'Contents' in response:
139             for obj in response['Contents']:
140                 source_key = obj['Key']
141                 archive_key = source_key.replace(source_prefix, archive_prefix)
142                 if not archive_key.startswith(archive_prefix):
143                     archive_key = archive_prefix + source_key.split('/')[-1]
144
145                 s3_c (function) CopySource: Any
146                     CopySource={'Bucket': bucket, 'Key': source_key},
147                     Bucket=bucket,
148                     Key=archive_key
149                 )
150                 s3_client.delete_object(Bucket=bucket, Key=source_key)
151                 logger.info(f"Moved files from {source_path} to {archive_path} successfully")
152     except Exception as e:
153         logging.error(f"Error files {source_path} to {archive_path} successfully")
154         raise Exception(f"Error files {source_path} to {archive_path} successfully")
155
156     try:
157         move_s3_files(orders_path, orders_archive_path)
158         move_s3_files(order_items_path, order_items_archive_path)
159         move_s3_files(products_path, products_archive_path)
160     except Exception as e:
161         logger.error(f"Error moving files to archived folders: {e}")

```

```

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```


docker-dynamo-etl-wrangler > app.py > ...


```
130 def move_s3_files(source_path, archive_path):
144
145     s3_client.copy_object(
146         CopySource={'Bucket': bucket, 'Key': source_key},
147         Bucket=bucket,
148         Key=archive_key
149     )
150     s3_client.delete_object(Bucket=bucket, Key=source_key)
151     logger.info(f"Moved files from {source_path} to {archive_path} successfully")
152 except Exception as e:
153     logging.error(f"Error files {source_path} to {archive_path} successfully")
154     raise Exception(f"Error files {source_path} to {archive_path} successfully")
155
156 try:
157     move_s3_files(orders_path, orders_archive_path)
158     move_s3_files(order_items_path, order_items_archive_path)
159     move_s3_files(products_path, products_archive_path)
160 except Exception as e:
161     logger.error(f"Error moving files to archived folders: {e}")
162     raise
```

```
130 def move_s3_files(source_path, archive_path):
144
145     s3_client.copy_object(
146         CopySource={'Bucket': bucket, 'Key': source_key},
147         Bucket=bucket,
148         Key=archive_key
149     )
150     s3_client.delete_object(Bucket=bucket, Key=source_key)
151     logger.info(f"Moved files from {source_path} to {archive_path} successfully")
152 except Exception as e:
153     logging.error(f"Error files {source_path} to {archive_path} successfully")
154     raise Exception(f"Error files {source_path} to {archive_path} successfully")
155
156 try:
157     move_s3_files(orders_path, orders_archive_path)
158     move_s3_files(order_items_path, order_items_archive_path)
159     move_s3_files(products_path, products_archive_path)
160 except Exception as e:
161     logger.error(f"Error moving files to archived folders: {e}")
162     raise
```


- ✓ ECS-STEP-PYTHON
 - > data
 - ✓ docker-data-validity
 - app.py
 - Dockerfile
 - requirements.txt
 - ✓ docker-dynamo-etl-wrangler
 - app.py
 - Dockerfile**
 - requirements.txt
 - ✓ step-functions
 - { } step-function.json
 - { } step-functions-iam-execution-pol...
 - \$ docker-commands.sh
 - { } event-pattern.json
 - test.ipynb

docker-dynamo-etl-wrangler > Dockerfile

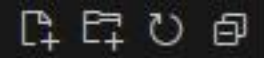
```
1 FROM python:3.10
2
3 WORKDIR /usr/src/app
4
5 COPY . .
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 CMD ["python3", "app.py"]
10
```

 Do you want to install the recommended 'Docker' extension from Microsoft for the Docker language?

Install

Show Recommendations

✓ ECS-STEP-PYTHON



- > data
- ✓ docker-data-validity
 - 🔗 app.py
 - 🔗 Dockerfile
 - ≡ requirements.txt

- ✓ docker-dynamo-etl-wrangler
 - 🔗 app.py
 - 🔗 Dockerfile

≡ requirements.txt

- ✓ step-functions
 - { } step-function.json
 - { } step-functions-iam-execution-pol...
 - \$ docker-commands.sh
 - { } event-pattern.json
 - 📘 test.ipynb

> OUTLINE
> TIMELINE

docker-dynamo-etl-wrangler > ≡ requirements.txt

```
1 pandas
2 awswrangler
```

Do you want to install the recommended 'Docker' extension from Microsoft for the Docker language?

Install

Show Recommendations

\$ docker-commands.sh

```
1  # Docker Authentication with ECR
2  aws ecr get-login-password \
3      --region us-east-1 | docker login \
4      --username AWS 418272783111\
5      --password-stdin .dkr.ecr.us-east-1.amazonaws.com
6
7  # Commands for data-validation task
8  docker build -t ecom_data_validation .
9  docker run -d -v ~/.aws:/root/.aws ecom_data_validation
10 docker tag ecom_data_validation:latest 418272783111.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:ecom_data_validation
11 docker push 418272783111.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:ecom_data_validation
12
13 # Newly tagged image
14 docker tag ecom_data_validation:latest {aws-account-number}.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:ecom_data_va
15 docker push {aws-account-number}.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:ecom_data_validation_v1
16
17
18 # Commands for ETL Job
19 docker build -t etl_aggregations .
20 docker run -d -v ~/.aws:/root/.aws etl_aggregations
21 docker tag etl_aggregations:latest 418272783111.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:etl_aggregations
22 docker push 418272783111.dkr.ecr.us-east-1.amazonaws.com/ecommerce-pipelines:etl_aggregations
23
24
25 # Get task definition once created
26 aws ecs describe-task-definition \
27     --task-definition redshift-ingestion \
28     --query taskDefinition "Angle Bracket" task-definition.json
29
30
31 aws ecs describe-task-definition --task-definition redshift-ingestion
```

ecommerce-data/

Copy S3 URI

Objects

Properties

Objects (2) Info



Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

< 1 > Settings

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	archived/	Folder	-	-	-
<input type="checkbox"/>	new/	Folder	-	-	-



<input type="checkbox"/>	Name	Event types	Filters	Destination type	Destination
--------------------------	------	-------------	---------	------------------	-------------

No event notifications

Choose **Create event notification** to be notified when a specific event occurs.

Create event notification

Amazon EventBridge

Edit

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. [Learn more](#) or [see EventBridge pricing](#)

Send notifications to Amazon EventBridge for all events in this bucket

On

Transfer acceleration

Edit

Use an accelerated endpoint for faster data transfers. [Learn more](#)

Transfer acceleration

Disabled

Object Lock

Edit

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. [Learn more](#)




Amazon Elastic Container Service


Clusters Updated


Namespaces


Task definitions


Account settings Updated


Install AWS Copilot 


Amazon ECR 

Repositories 

AWS Batch 

Documentation 

Discover products 

Subscriptions 

Task definitions (2) Info

Last updated
27 December 2024 at 14:30 (UTC+5:30) 

Deploy ▼

Create new revision ▼

Create new task definition ▼

 *Filter task definitions*

Filter by status
Active ▼

< 1 > 

	Task definition ▼	Status of last revision ▼
<input type="radio"/>	task-data-validation	✓ ACTIVE
<input type="radio"/>	task-etl-calculations	✓ ACTIVE





Amazon Elastic
Container Service

Clusters [Updated](#)

Namespaces

Task definitions

Account settings [Updated](#)

Install AWS Copilot [↗](#)

Amazon ECR [↗](#)

Repositories [↗](#)

AWS Batch [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Subscriptions [↗](#)

task-data-validation:4


Deploy ▼

Actions ▼

Create new revision ▼

Overview [Info](#)

ARN

 arn:aws:ecs:us-east-1:418272783111:task-definition/task-data-validation:4

Status

 **ACTIVE**

Time created

27 December 2024 at 13:22 (UTC+5:30)

App environment

Fargate

Task role

[custom-ecs-task-execution-role](#) [↗](#)

Task execution role

[custom-ecs-task-execution-role](#) [↗](#)

Operating system/Architecture
Linux/X86_64

Network mode

awsvpc

Fault injection

 Turned off

Containers

JSON

Task placement

Volumes (0)

Requires attributes

Tags

Task size

Task CPU

2,048 units (2 vCPU)

Task memory

8,192 MiB (8 GB)





Amazon Elastic Container Service

- Clusters [Updated](#)
- Namespaces
- Task definitions**
- Account settings [Updated](#)

[Install AWS Copilot](#)

[Amazon ECR](#)

[Repositories](#)

[AWS Batch](#)

[Documentation](#)

[Discover products](#)

[Subscriptions](#)

Containers

JSON

Task placement

Volumes (0)

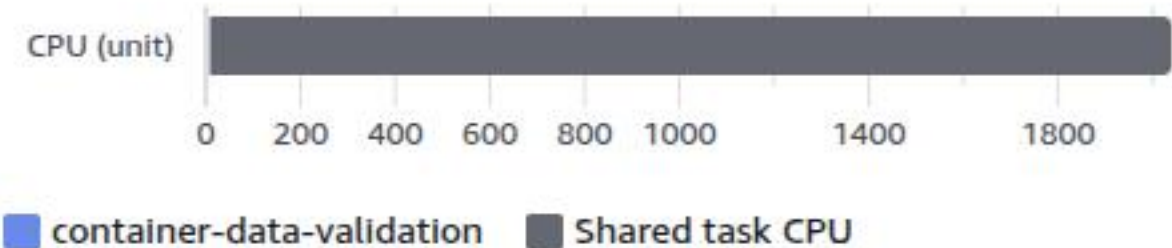
Requires attributes

Tags

Task size

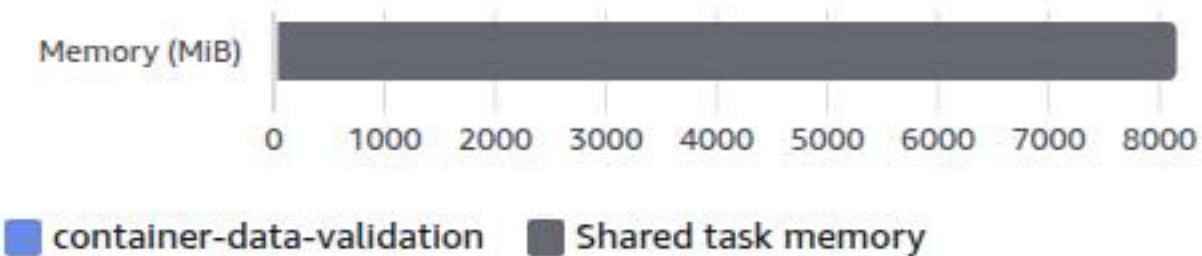
Task CPU
2,048 units (2 vCPU)

Task CPU maximum allocation for containers



Task memory
8,192 MiB (8 GB)

Task memory maximum allocation for container memory reservation



Containers [Info](#)

Container name	Image	Private reg...	Essential	CPU	Memory ha...	GPU
container-dat...	41827...	-	Yes	0	-/-	-





Amazon Elastic Container Service

- Clusters [Updated](#)
- Namespaces
- Task definitions**
- Account settings [Updated](#)

[Install AWS Copilot](#)

[Amazon ECR](#)
[Repositories](#)

[AWS Batch](#)

[Documentation](#)
[Discover products](#)
[Subscriptions](#)

task-etl-calculations:3

Deploy ▼

Actions ▼

Create new revision ▼

Overview [Info](#)

ARN arn:aws:ecs:us-east-1:418272783111:task-definition/task-etl-calculations:3	Status ACTIVE	Time created 27 December 2024 at 12:43 (UTC+5:30)	App environment Fargate
Task role custom-ecs-task-execution-role	Task execution role custom-ecs-task-execution-role	Operating system/Architecture Linux/X86_64	Network mode awsvpc
Fault injection Turned off			

- Containers
- JSON
- Task placement
- Volumes (0)
- Requires attributes
- Tags

Task size

Task CPU 2,048 units (2 vCPU)	Task memory 10,240 MiB (10 GB)
---	--





Amazon Elastic Container Service

- Clusters [Updated](#)
- Namespaces
- Task definitions**
- Account settings [Updated](#)

[Install AWS Copilot](#)

[Amazon ECR](#)

[Repositories](#)

[AWS Batch](#)

[Documentation](#)

[Discover products](#)

[Subscriptions](#)

Containers

JSON

Task placement

Volumes (0)

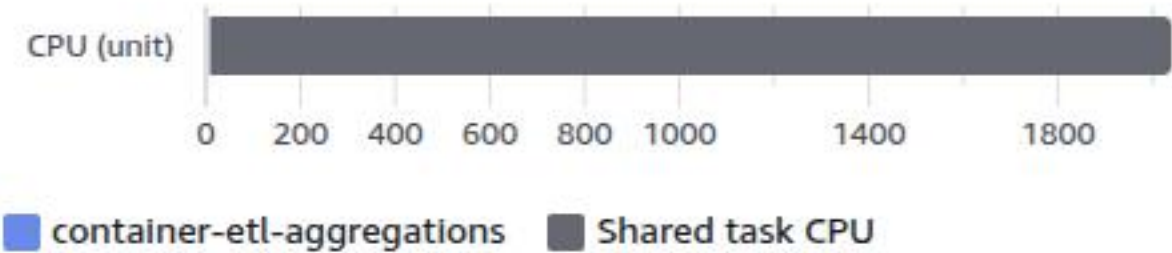
Requires attributes

Tags

Task size

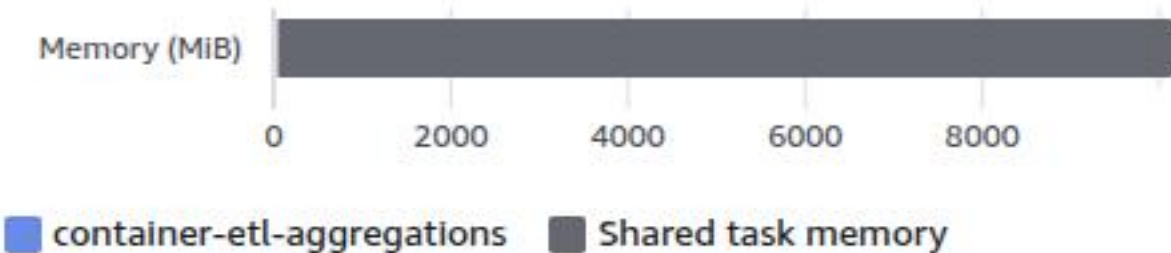
Task CPU
2,048 units (2 vCPU)

Task CPU maximum allocation for containers



Task memory
10,240 MiB (10 GB)

Task memory maximum allocation for container memory reservation



Containers [Info](#)

Container name	Image	Private reg...	Essential	CPU	Memory ha...	GPU
container-etl-...	41827...	-	Yes	0	-/-	-





DynamoDB



Dashboard

Tables

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations [New](#)

Reserved capacity

Settings

▼ DAX

Clusters

Subnet groups

Parameter groups

Events

PartiQL editor

Operations performed using the PartiQL editor might incur charges. [Learn more](#)

Tables (2)



< 1 > ⚙

▶ category_wise_summary ...

▼ daily_order_summary ...

order_date ...
Partition key

Query 1



1 select * from daily_order_summary;

Run

Clear





DynamoDB



- Dashboard
- Tables
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations New
- Reserved capacity
- Settings

DAX

- Clusters
- Subnet groups
- Parameter groups
- Events

Run Clear

Table view JSON view

Completed
Started on 27/12/2024, 16:18:49
Elapsed time 268ms

Items returned (2) Download results to CSV

Find items

< 1 > Settings

order_date	unique_customers	return_rate	total_revenue
2024-06-01	387	0.106965174...	56816.6200499...
2024-06-02	366	0.109947643...	63619.1099960...

