Ingestion and historization in the Data Lake

Illia Todor

/iamtodor

Data Engineer

AWS Cloud Practitioner Certified

/iamtodor

Data Engineer

AWS Cloud Practitioner Certified

HRS GROUP

/iamtodor

Data Engineer

AWS Cloud Practitioner Certified

HRS GROUP

https://github.com/iamtodor

https://stackoverflow.com/users/5151861/iamtodor

https://iamtodor.medium.com/

https://www.linkedin.com/in/iamtodor/

HRS

HRS GROUP

provides the world's premier lodging experience to corporations globally.

50
markets worldwide

32,000
cities globally

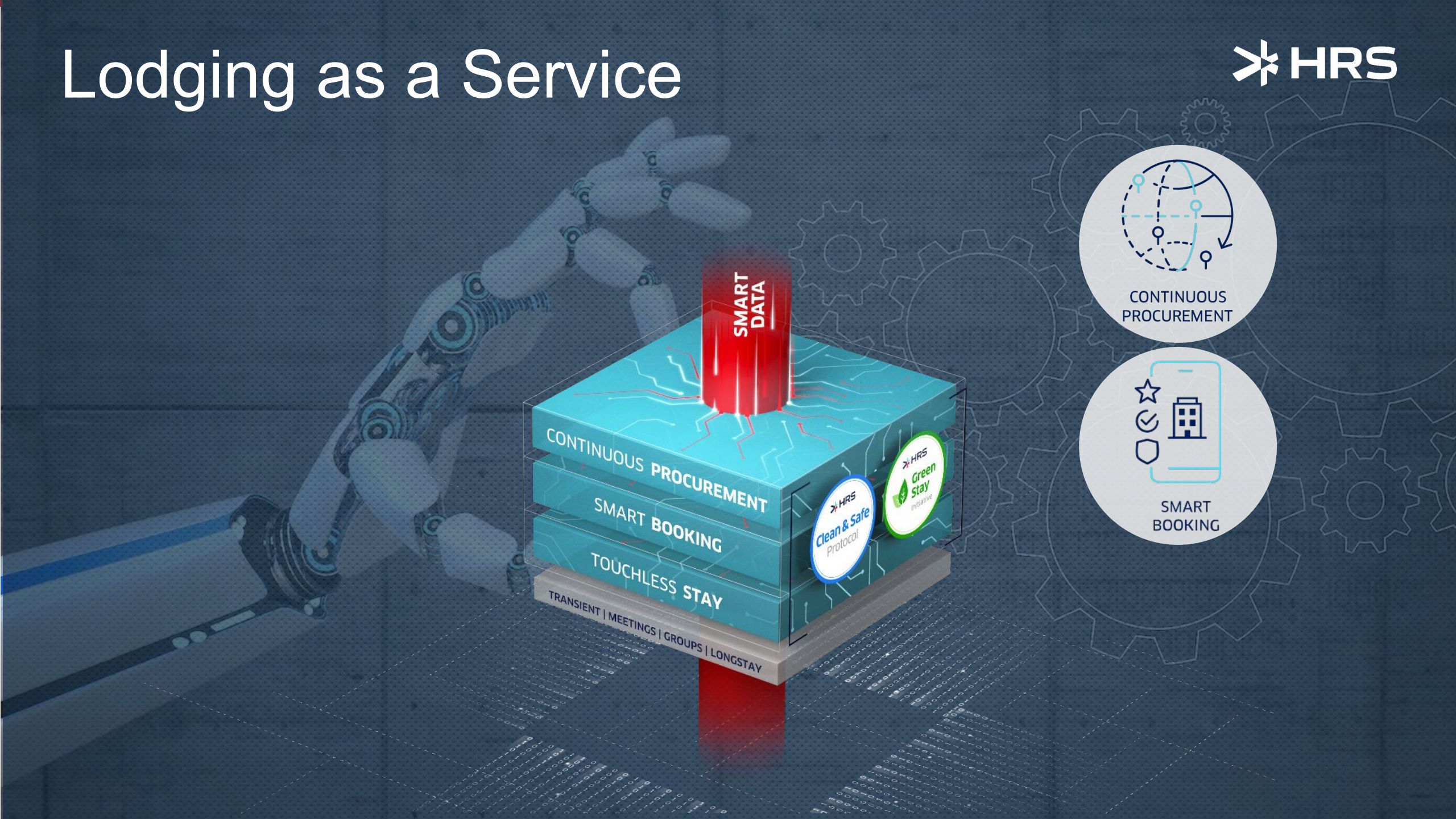
1,000
employees

> 800.000
yearly contacts with Hotels

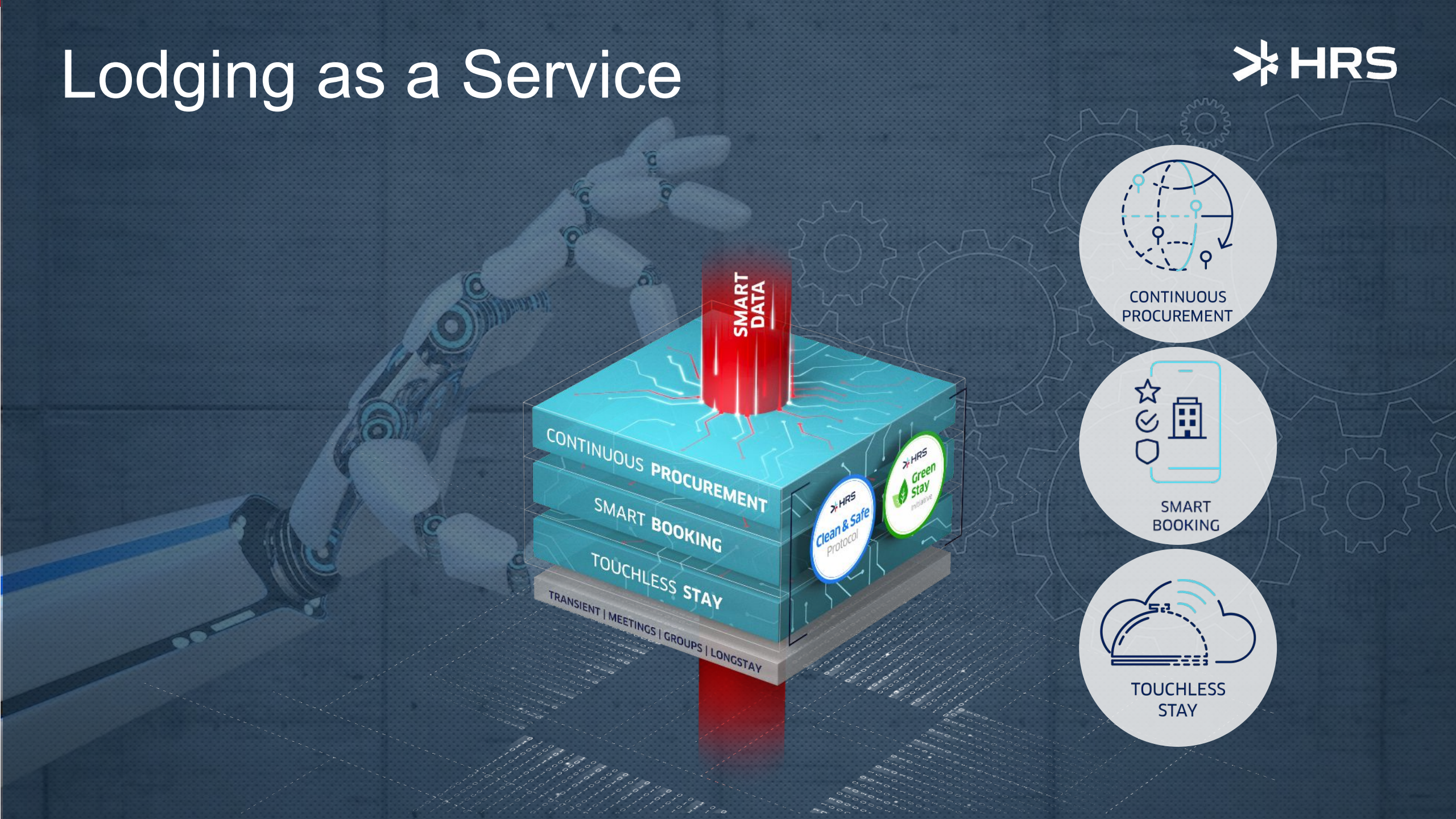
> 6,000
Global customers

Lodging as a Service

Lodging as a Service



Lodging as a Service



Request

- DataSource
- Historical Changes
- DataLake
- DataWarehouse
- Dashboards

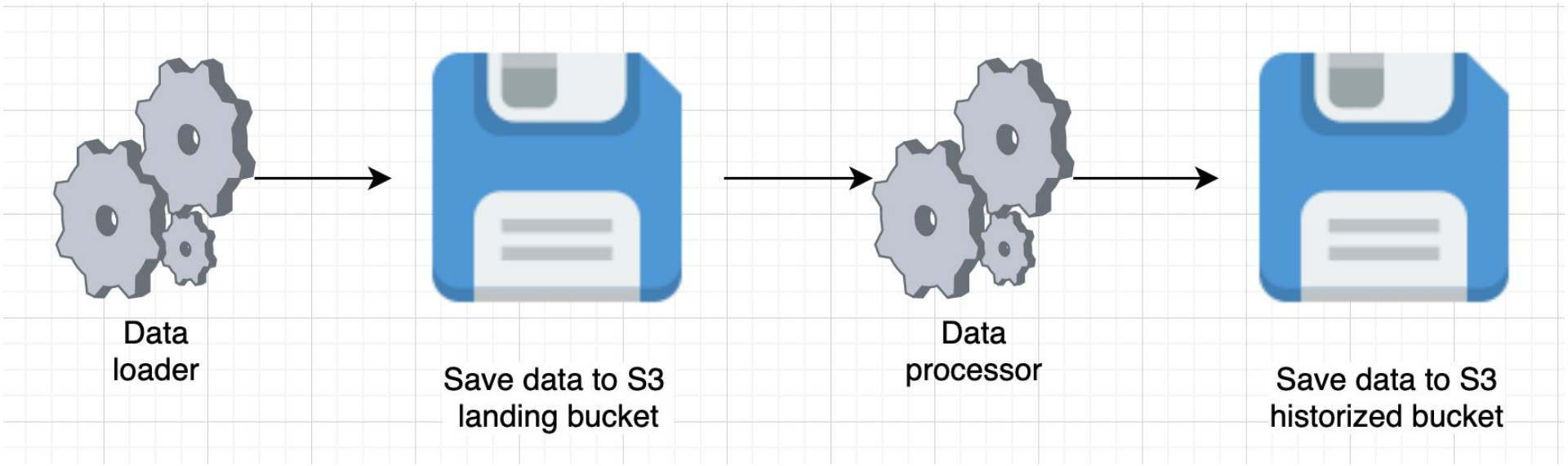
Historization: the process

We wanted to move ingestions from DataSource into Data Lake along with applying historization approach. The data we need in the DWH for reporting.

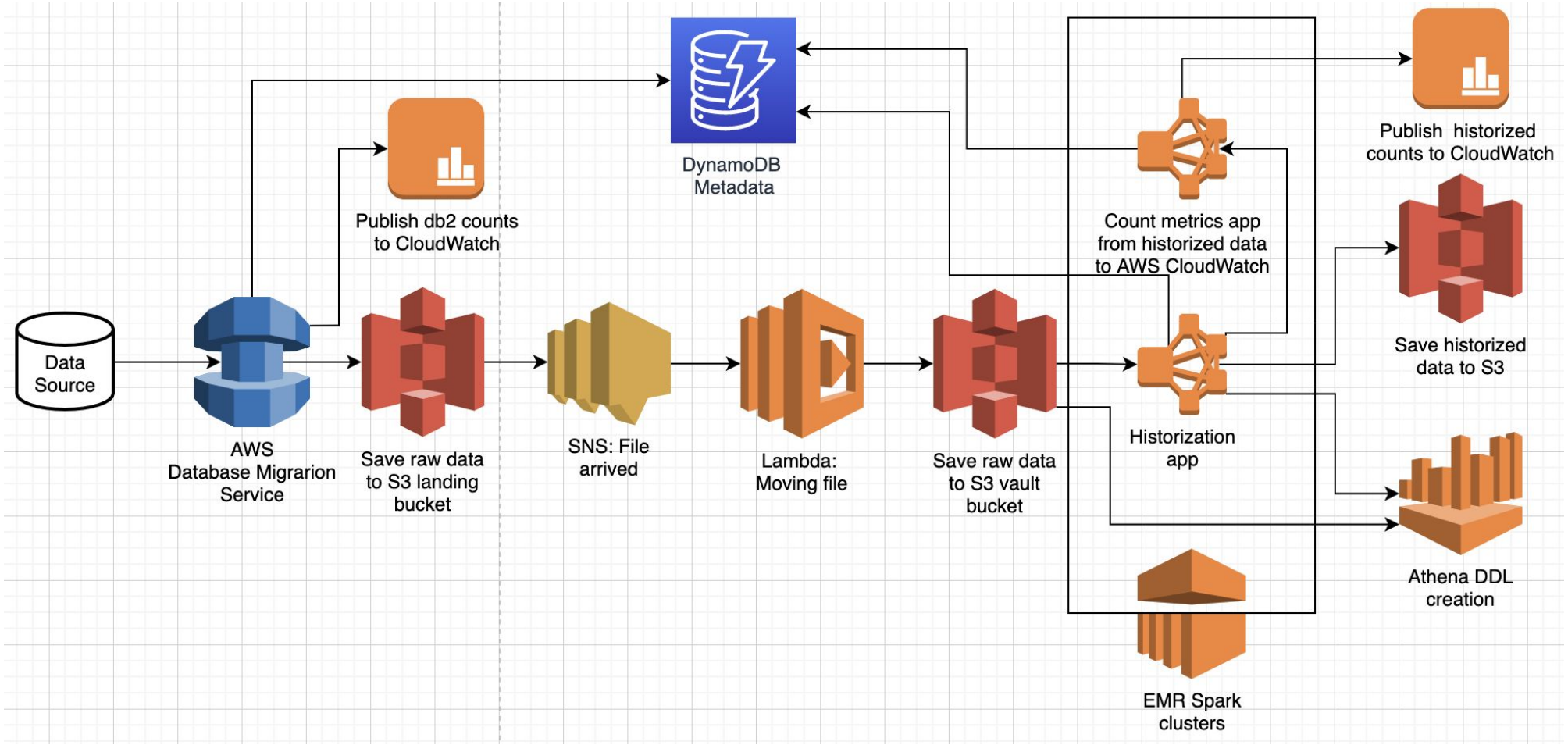
The idea of historization approach is to track historical changes.

Operation in the source DB	Field value	valid_from	valid_to
Insert	Andrew	2021-01-01	2021-01-03
Update	Mark	2021-01-03	2021-01-05
Update	Daniel	2021-01-05	2099-12-31

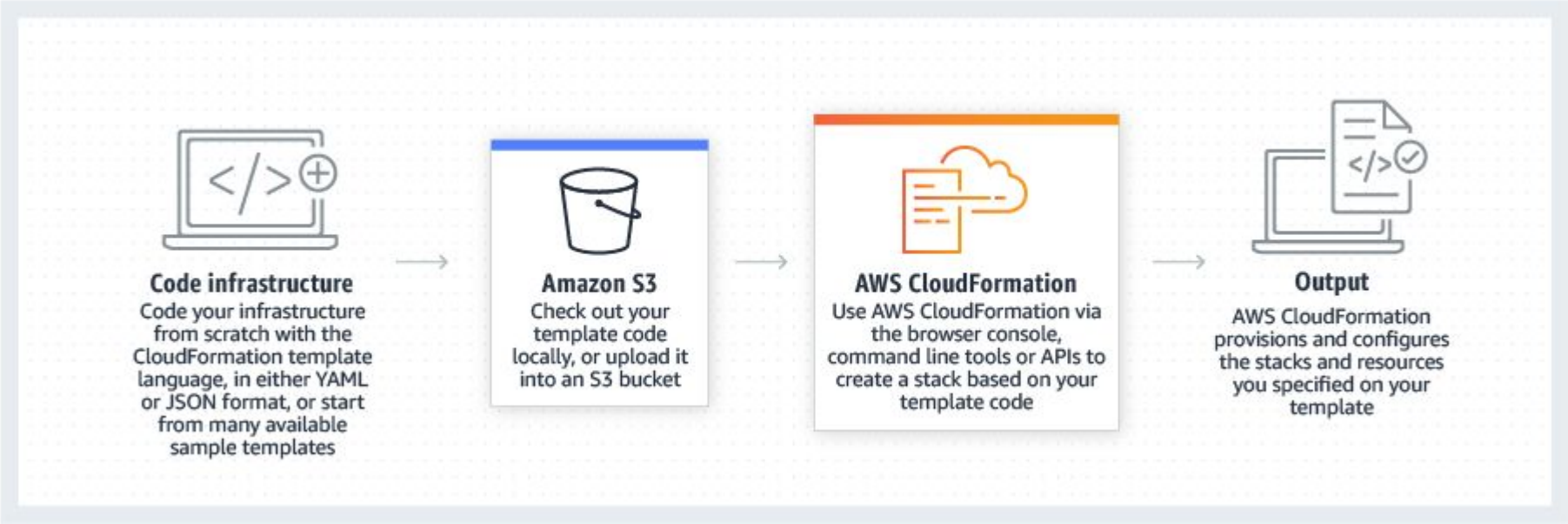
Architecture: abstract



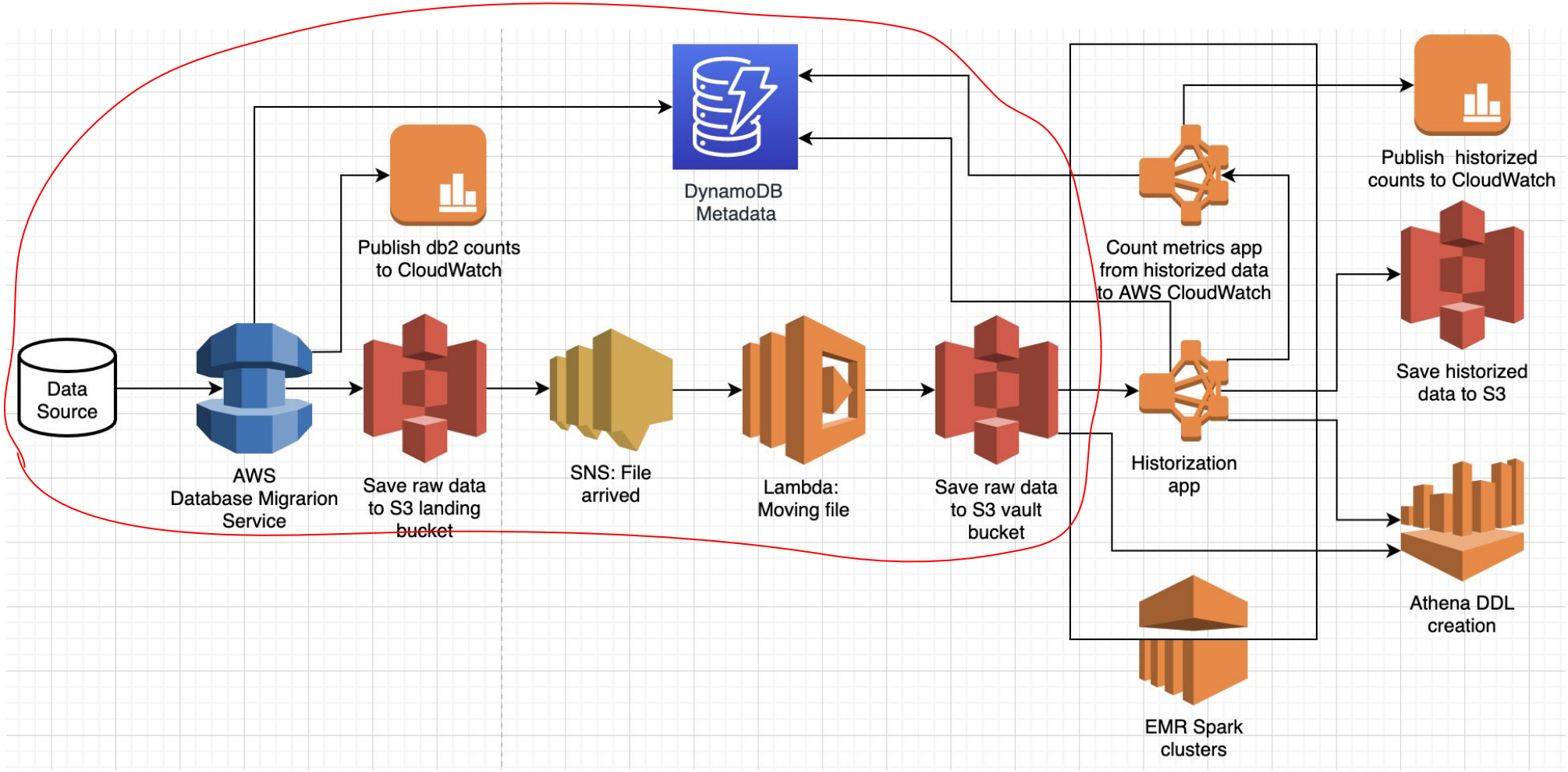
Architecture: in details



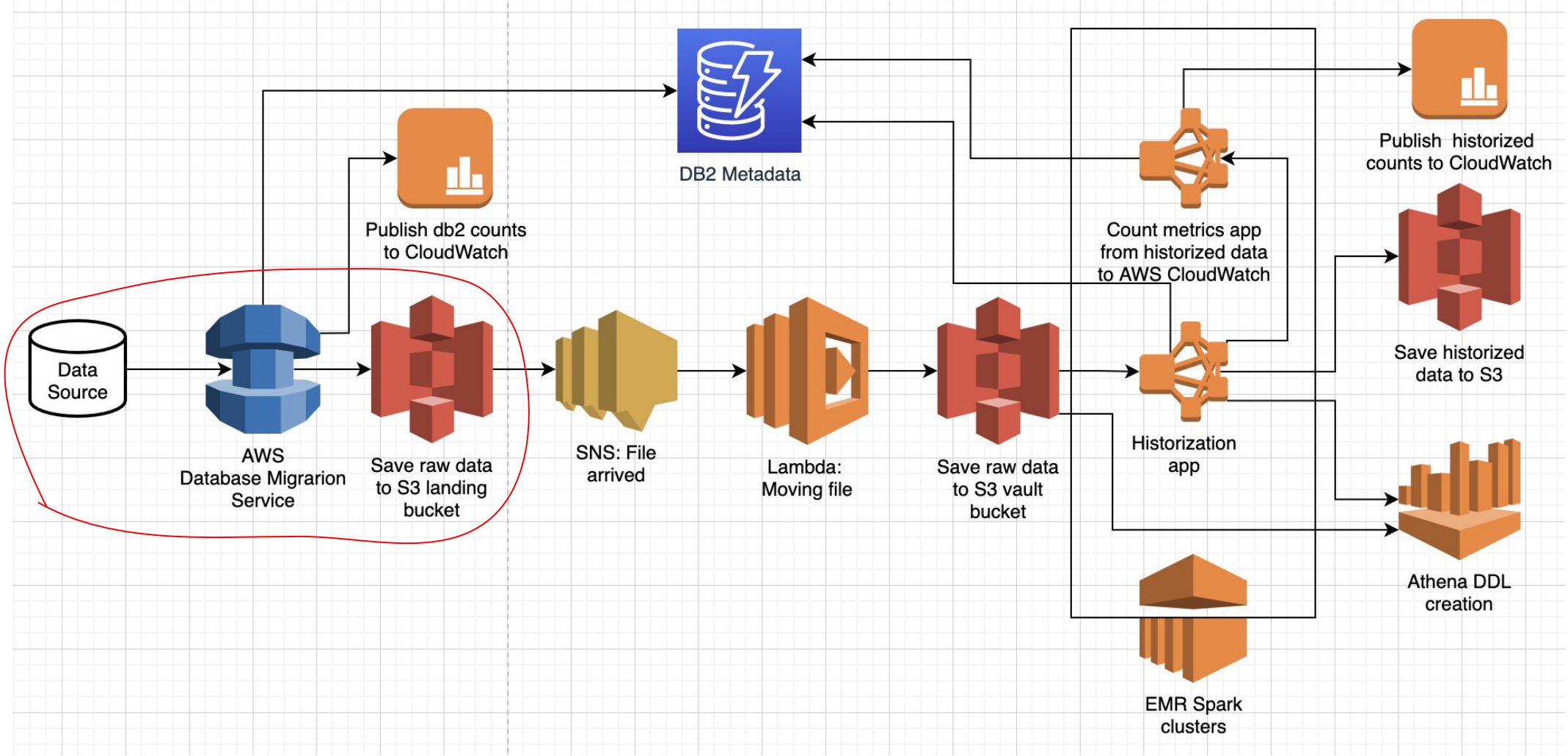
Infrastructure as a Code



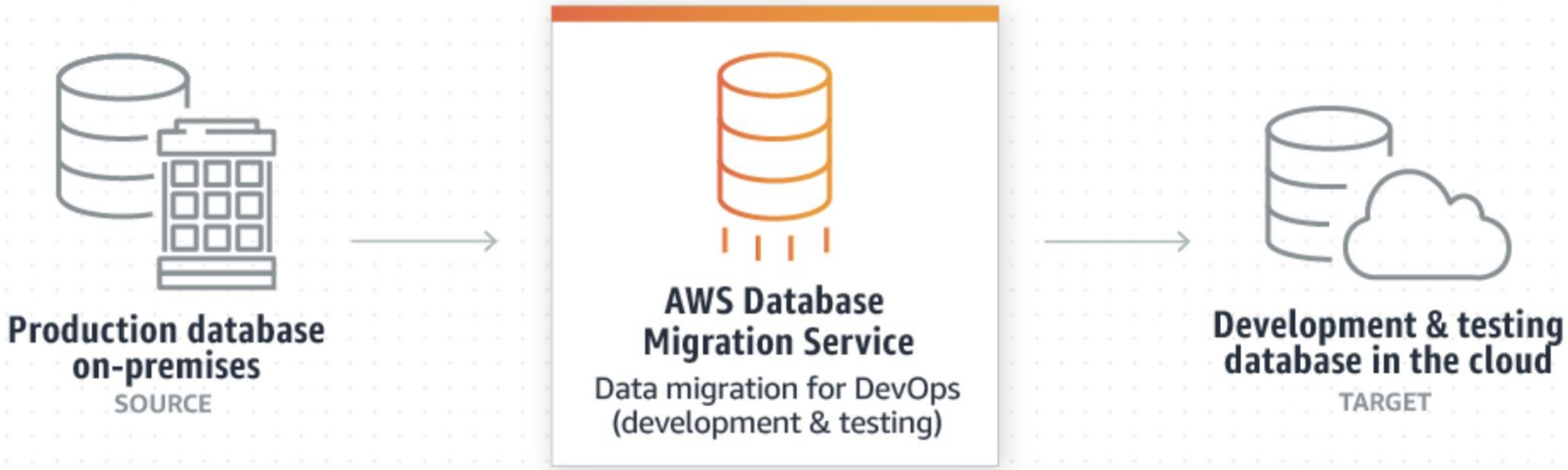
First part: global overview



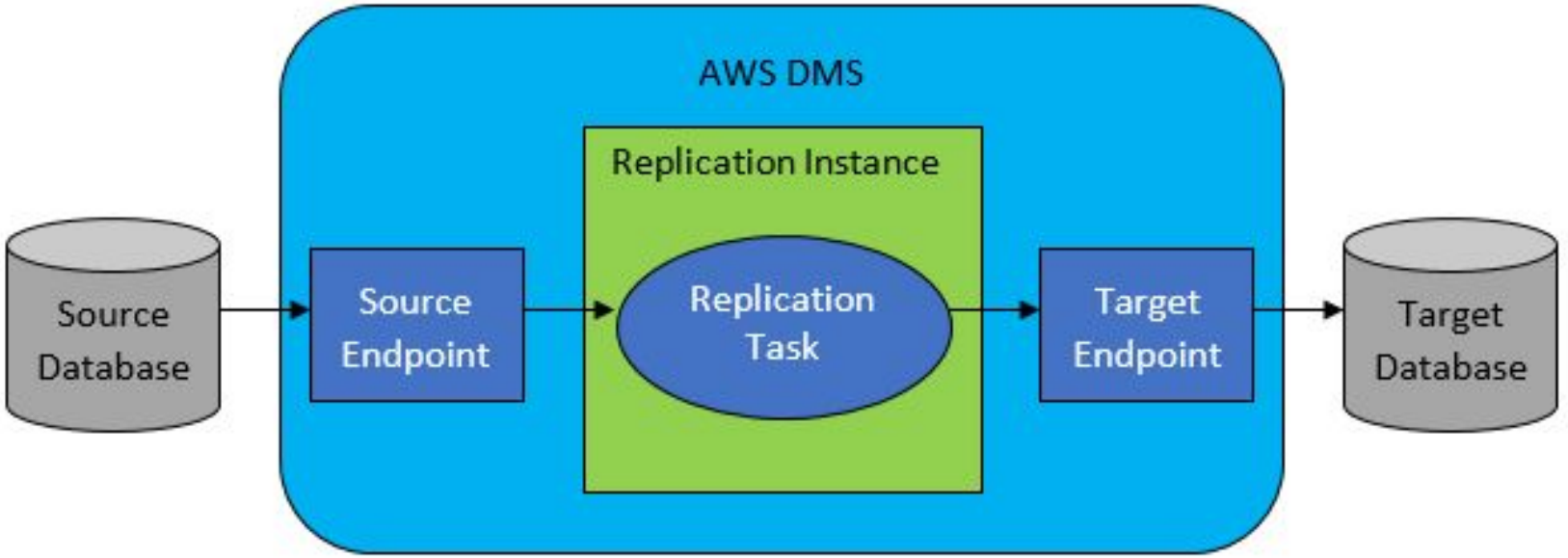
First part: DMS



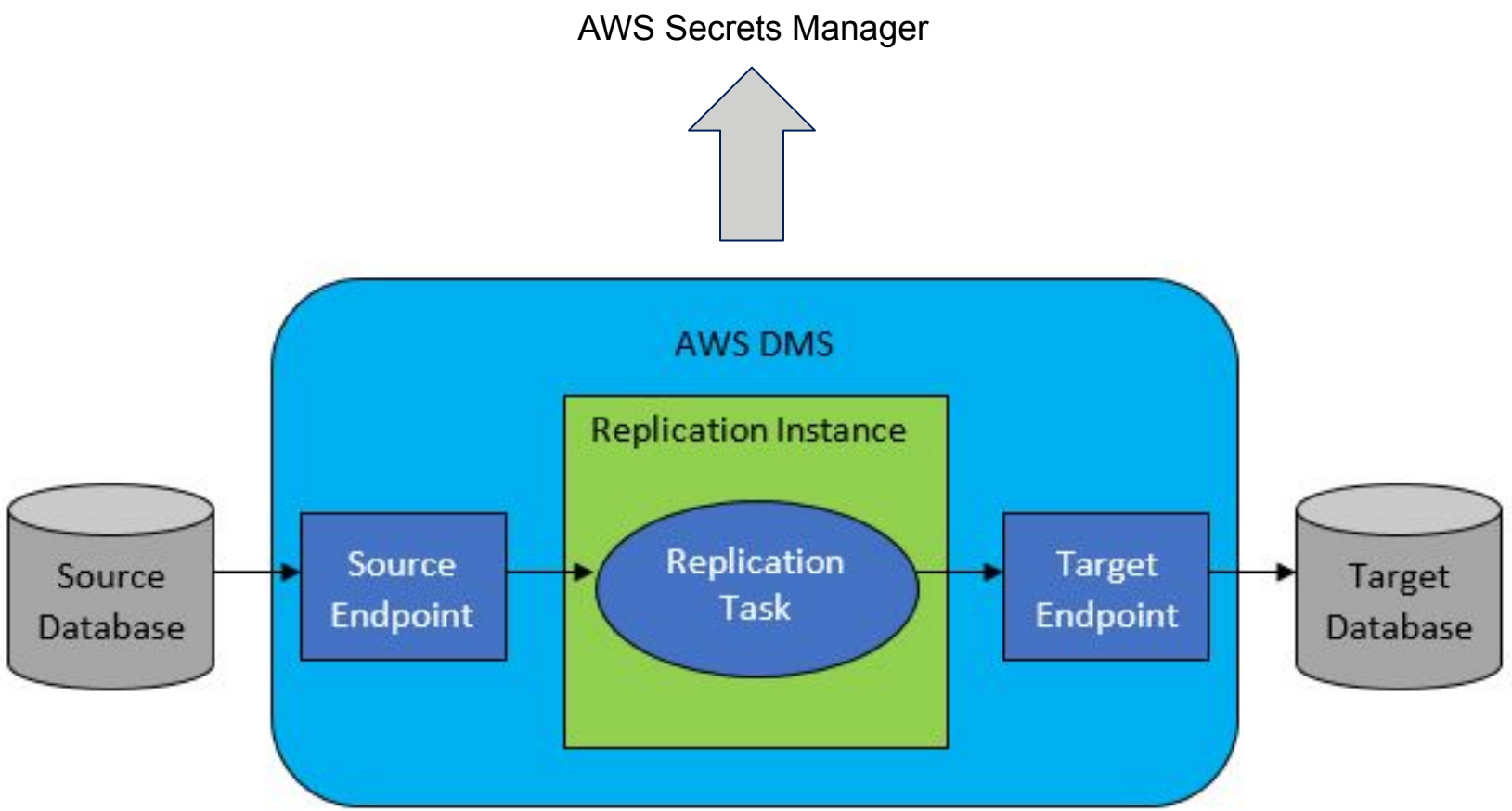
AWS DMS: Database Migration Service



AWS DMS: Database Migration Service



AWS DMS: Database Migration Service



DMS: Replication Instance

Replication instance type	vCPU	Memory (GiB)
Memory Optimized		
dms.r4.large	2	15.25
dms.r4.xlarge	4	30.5
dms.r4.2xlarge	8	61
dms.r4.4xlarge	16	122
dms.r4.8xlarge	32	244
dms.r5.large	2	16
dms.r5.xlarge	4	32
dms.r5.2xlarge	8	64
dms.r5.4xlarge	16	128

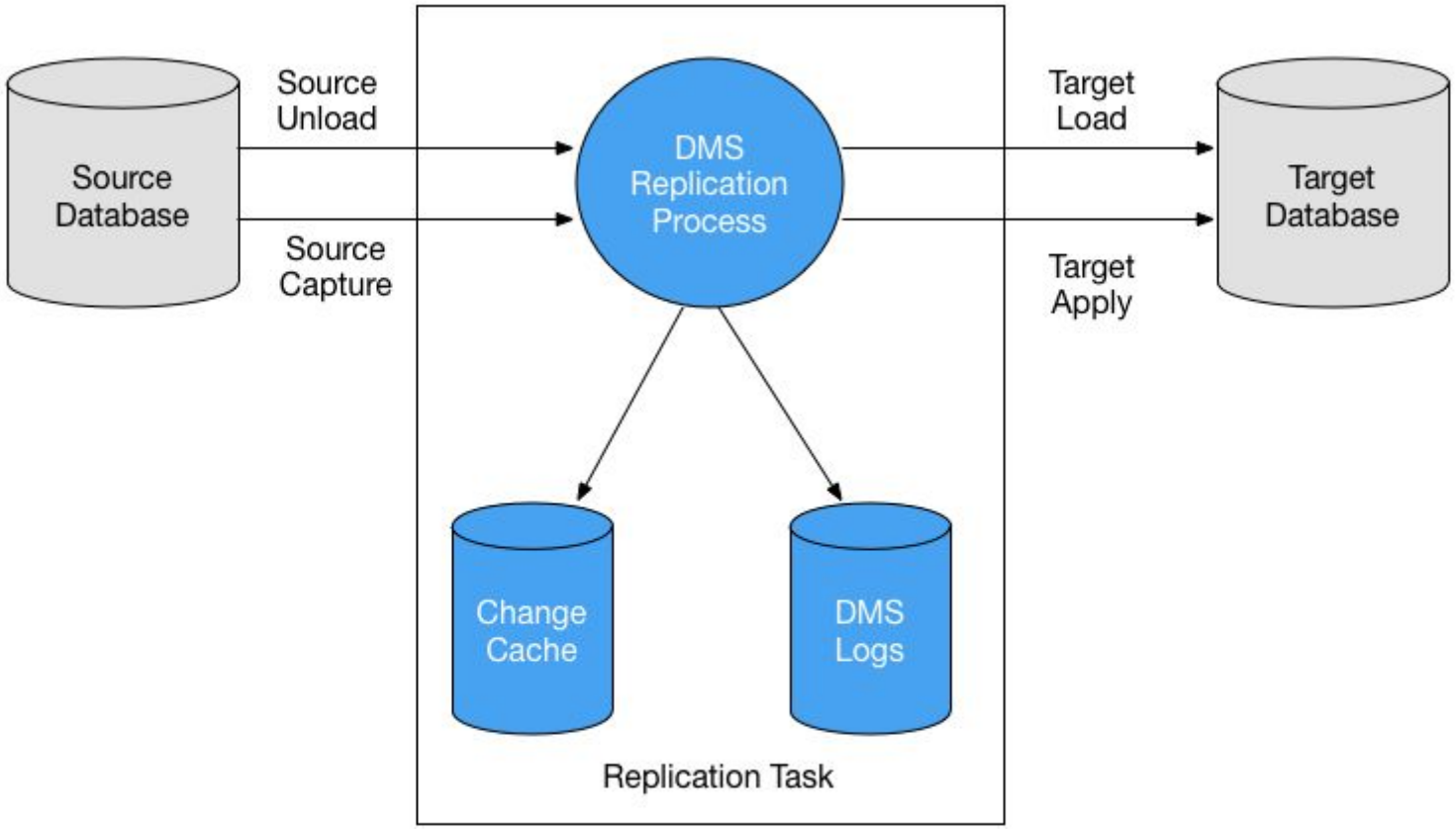
DMS: Target Endpoint

```
S3ProdDMSEndpoint:
  Type: AWS::DMS::Endpoint
  Properties:
    EndpointIdentifier: prod-s3-landing-parquet-endpoint
    EngineName: s3
    EndpointType: target
    S3Settings:
      BucketName: 'bucket_name'
      BucketFolder: 'prod/prefix'
      CompressionType: gzip
    ExtraConnectionAttributes:
'dataFormat=parquet;datePartitionDelimiter=DASH;datePartitionEnabled=false;datePartitionSequence=YYYYMMDD;'
```

DMS: Target Endpoint

```yaml
S3ProdDMSEndpoint:
  Type: AWS::DMS::Endpoint
  Properties:
    EndpointIdentifier: prod-s3-landing-parquet-endpoint
    EngineName: s3
    EndpointType: target
    S3Settings:
      BucketName: 'bucket_name'
      BucketFolder: 'prod/prefix'
      CompressionType: gzip
    ExtraConnectionAttributes:
'dataFormat=parquet;datePartitionDelimiter=DASH;datePartitionEnabled=false;datePartitionSequence=YYYYMMDD;'
```

DMS: Replication Task



DMS: Replication Task

```
ReplicationTask:
  Type: AWS::DMS::ReplicationTask
  Properties:
    ReplicationTaskIdentifier: !Sub "${Env}-{source}-{target}-{schema_name}-{table_name}"
    MigrationType: full-load
    ReplicationInstanceArn: !ImportValue replication-instance-arn
    SourceEndpointArn: !ImportValue source-endpoint-arn
    TargetEndpointArn: !ImportValue s3-prod-endpoint
    ReplicationTaskSettings: '{"Logging": {"EnableLogging": true}}'
    TableMappings: |
      {
        "rules": [
          {
            "rule-type": "selection",
            "rule-id": 1,
            "rule-name": 1,
            "rule-action": "include",
            "object-locator": {
              "schema-name": "schema_name",
              "table-name": "table_name"
            }
          }
        ]
      }
```

DMS: Replication Task

```
ReplicationTask:
  Type: AWS::DMS::ReplicationTask
  Properties:
    ReplicationTaskIdentifier: !Sub "${Env}-{source}-{target}-{schema_name}-{table_name}"
    MigrationType: full-load
    ReplicationInstanceArn: !ImportValue replication-instance-arn
    SourceEndpointArn: !ImportValue source-endpoint-arn
    TargetEndpointArn: !ImportValue s3-prod-endpoint
    ReplicationTaskSettings: '{"Logging": {"EnableLogging": true}}'
    TableMappings: |
      {
        "rules": [
          {
            "rule-type": "selection",
            "rule-id": 1,
            "rule-name": 1,
            "rule-action": "include",
            "object-locator": {
              "schema-name": "schema_name",
              "table-name": "table_name"
            }
          }
        ]
      }
```

DMS: Replication Task

```
ReplicationTask:
  Type: AWS::DMS::ReplicationTask
  Properties:
    ReplicationTaskIdentifier: !Sub "${Env}-{source}-{target}-{schema_name}-{table_name}"
    MigrationType: full-load
    ReplicationInstanceArn: !ImportValue replication-instance-arn
    SourceEndpointArn: !ImportValue source-endpoint-arn
    TargetEndpointArn: !ImportValue s3-prod-endpoint
    ReplicationTaskSettings: '{"Logging": {"EnableLogging": true}}'
    TableMappings: |
      {
        "rules": [
          {
            "rule-type": "selection",
            "rule-id": 1,
            "rule-name": 1,
            "rule-action": "include",
            "object-locator": {
              "schema-name": "schema_name",
              "table-name": "table_name"
            }
          }
        ]
      }
```

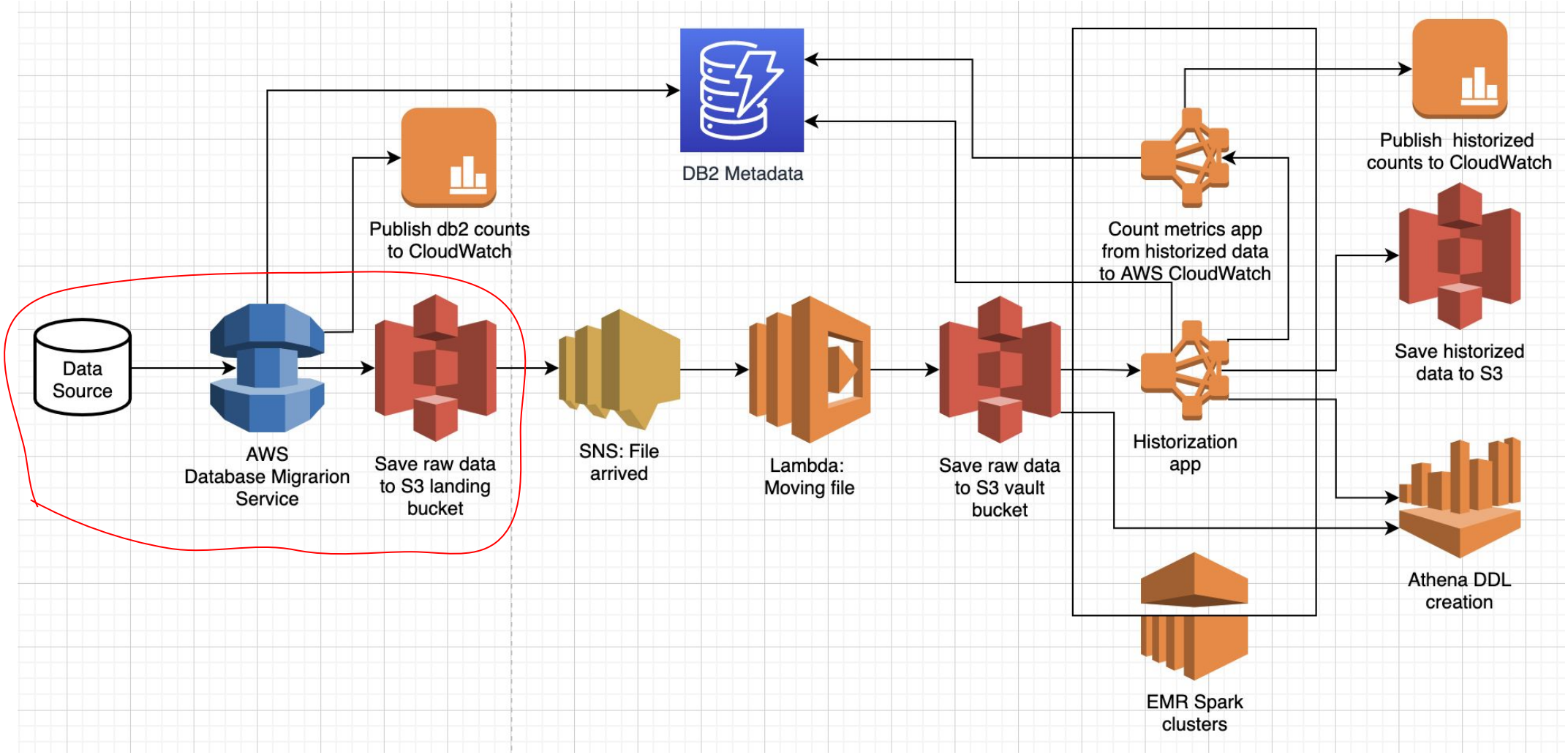
DMS: Replication Task

```
ReplicationTask:
  Type: AWS::DMS::ReplicationTask
  Properties:
    ReplicationTaskIdentifier: !Sub "${Env}-{source}-{target}-{schema_name}-{table_name}"
    MigrationType: full-load
    ReplicationInstanceArn: !ImportValue replication-instance-arn
    SourceEndpointArn: !ImportValue source-endpoint-arn
    TargetEndpointArn: !ImportValue s3-prod-endpoint
    ReplicationTaskSettings: '{"Logging": {"EnableLogging": true}}'
    TableMappings: |
      {
        "rules": [
          {
            "rule-type": "selection",
            "rule-id": 1,
            "rule-name": 1,
            "rule-action": "include",
            "object-locator": {
              "schema-name": "schema_name",
              "table-name": "table_name"
            }
          }
        ]
      }
```

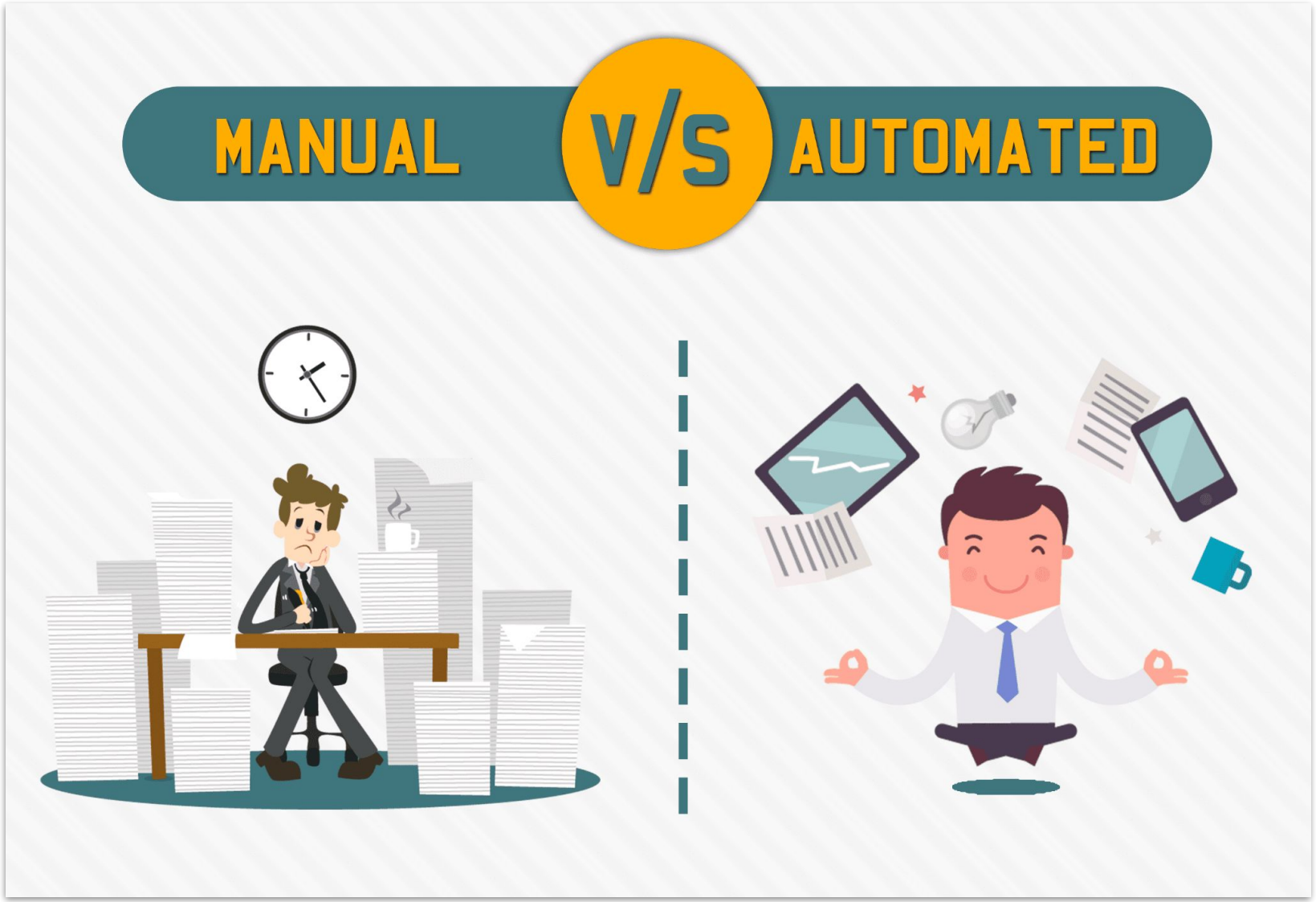
DMS: Replication Task

```
ReplicationTask:
  Type: AWS::DMS::ReplicationTask
  Properties:
    ReplicationTaskIdentifier: !Sub "${Env}-{source}-{target}-{schema_name}-{table_name}"
    MigrationType: full-load
    ReplicationInstanceArn: !ImportValue replication-instance-arn
    SourceEndpointArn: !ImportValue source-endpoint-arn
    TargetEndpointArn: !ImportValue s3-prod-endpoint
    ReplicationTaskSettings: '{"Logging": {"EnableLogging": true}}'
    TableMappings: |
      {
        "rules": [
          {
            "rule-type": "selection",
            "rule-id": 1,
            "rule-name": 1,
            "rule-action": "include",
            "object-locator": {
              "schema-name": "schema_name",
              "table-name": "table_name"
            }
          }
        ]
      }
```

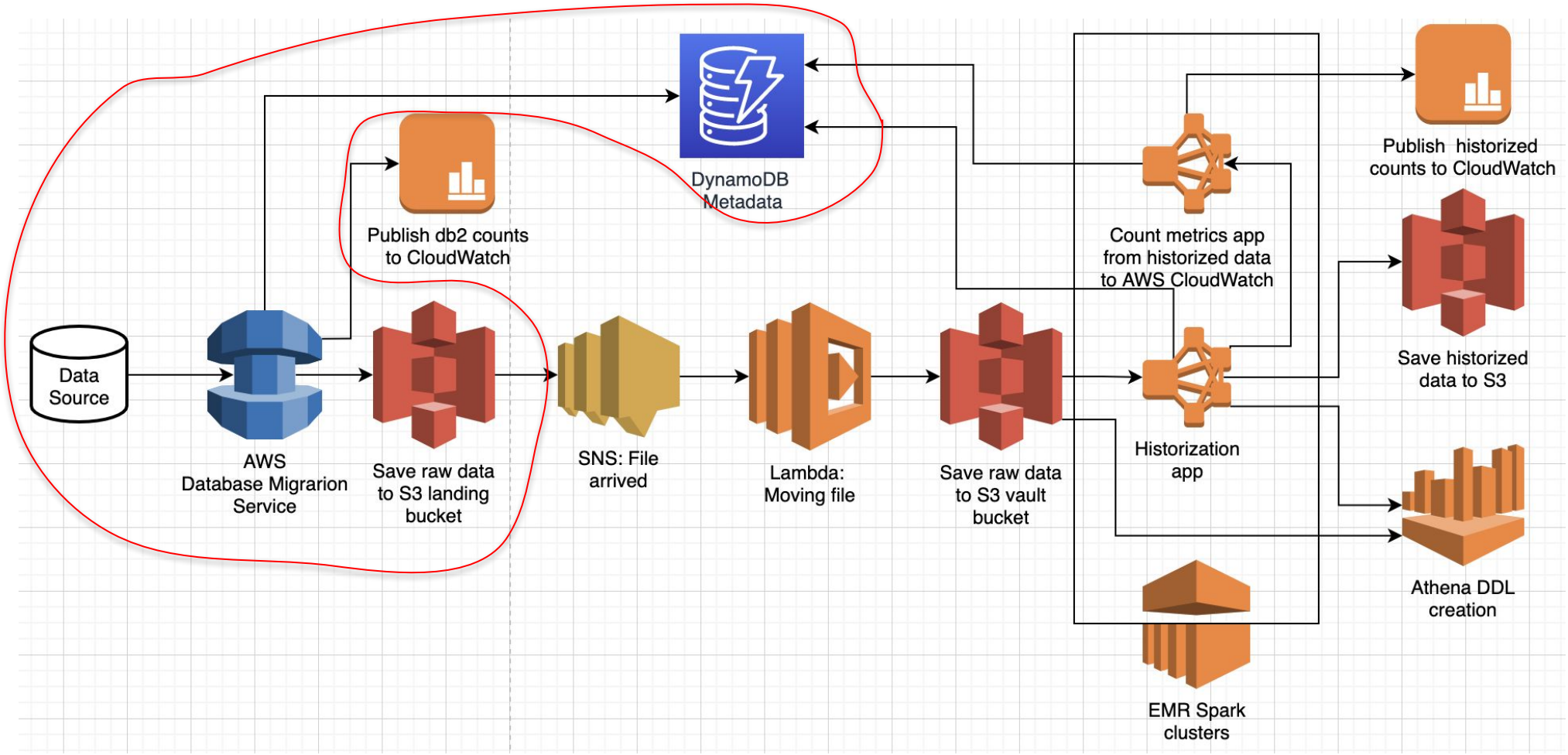
First part: DMS



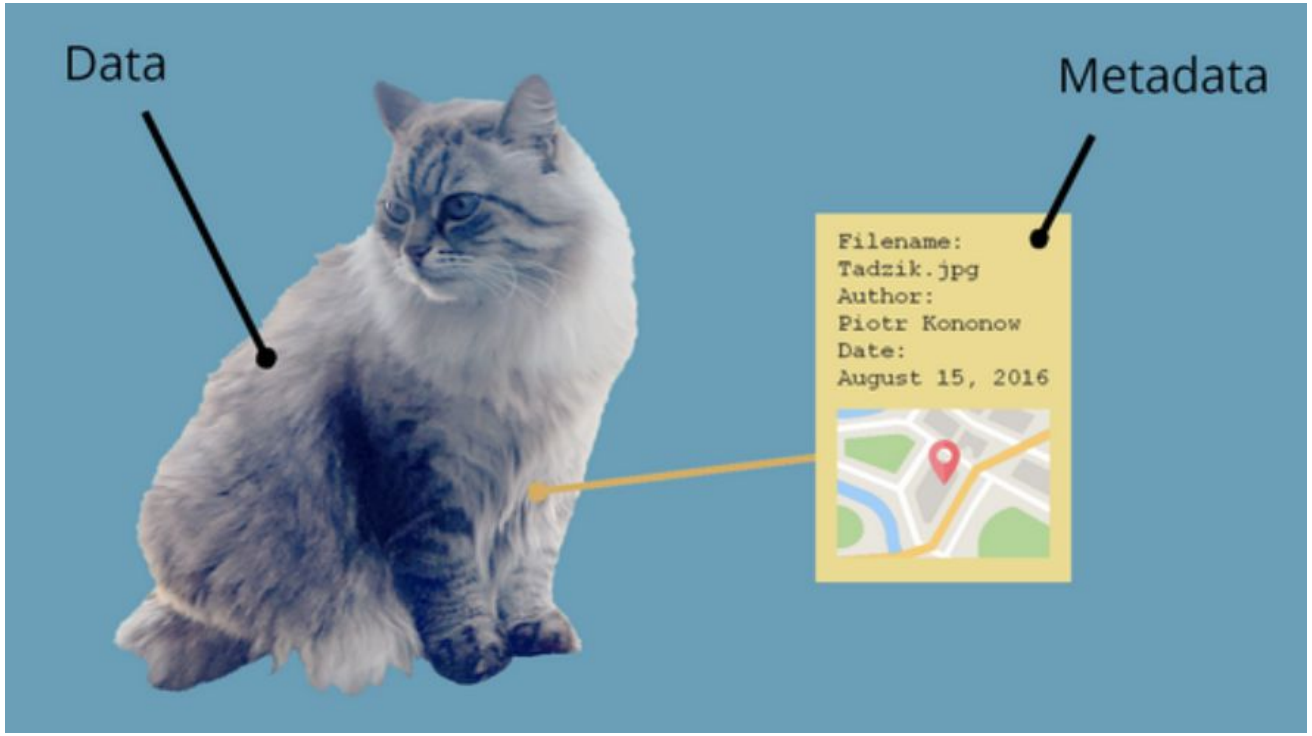
Process automation and scaling



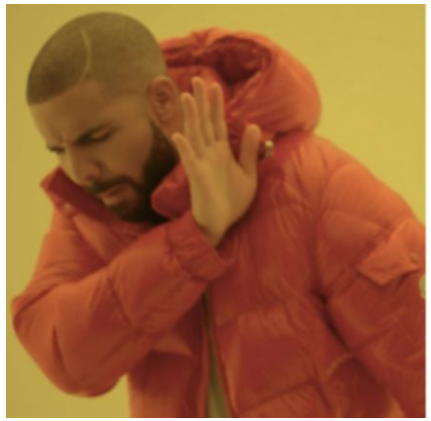
First part: Metadata management



What is metadata itself

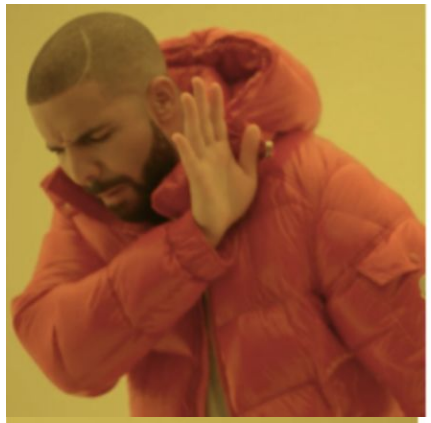


Suitable tool for metadata

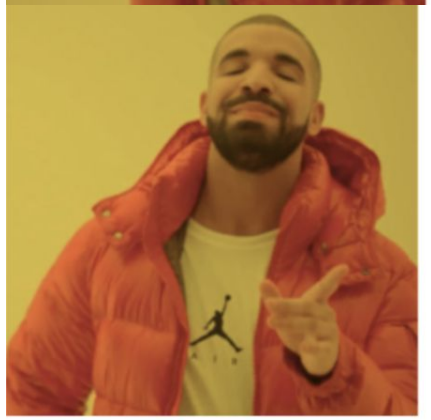


CSV

Suitable tool for metadata

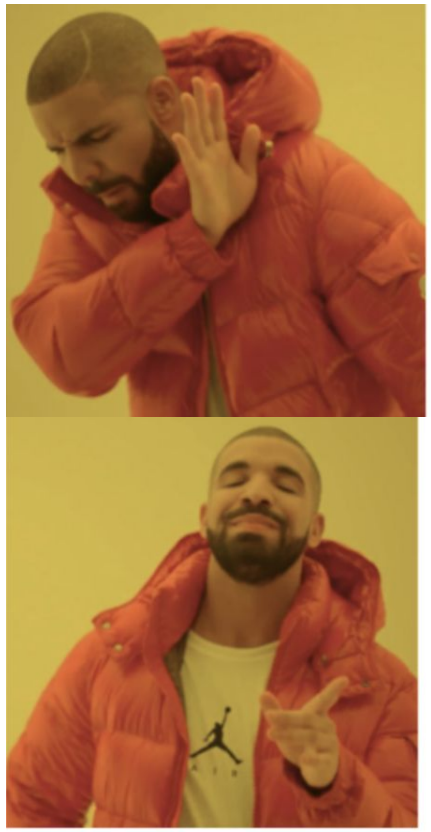


CSV



DynamoDB

Suitable tool for metadata



DynamoDB

amazon web services™

config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

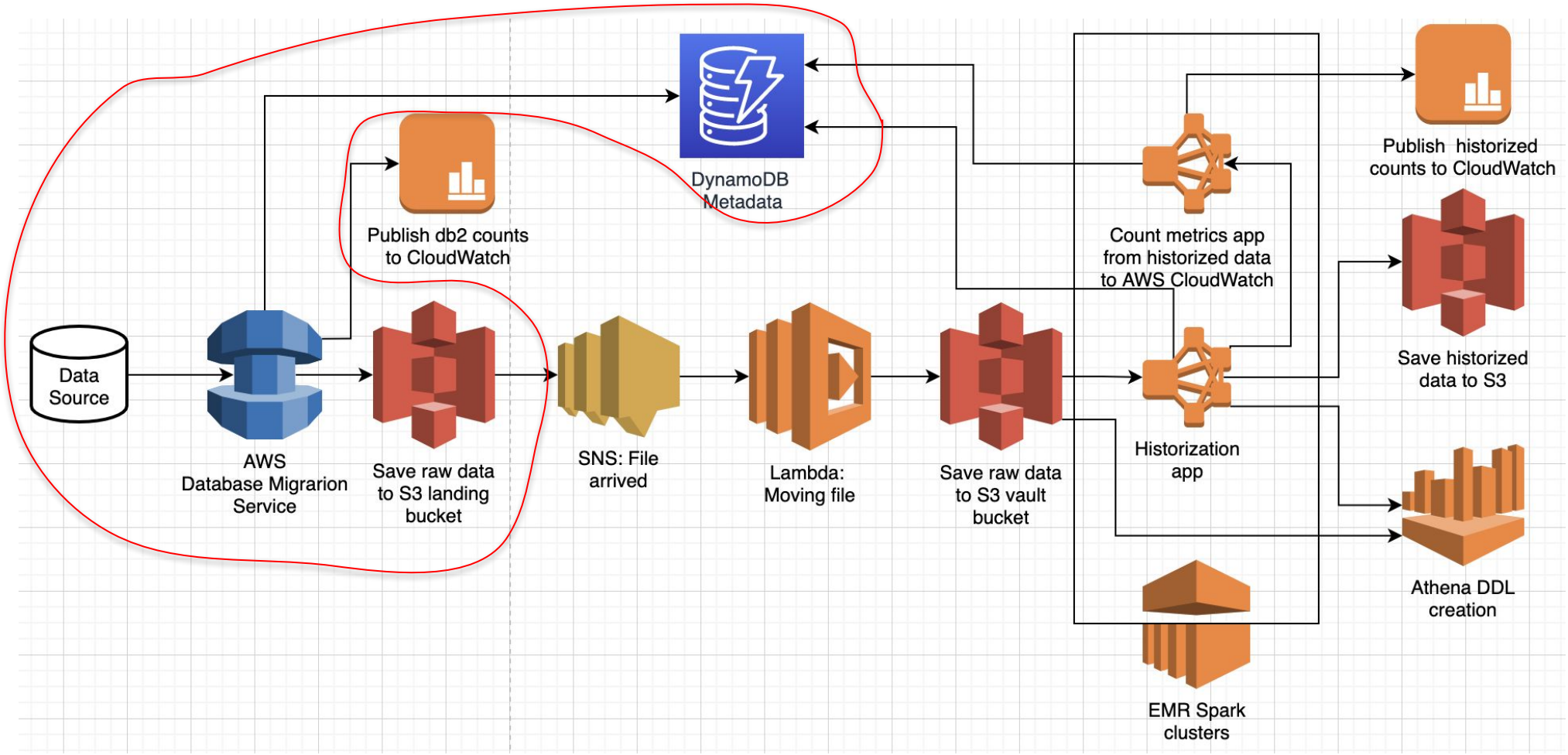
config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

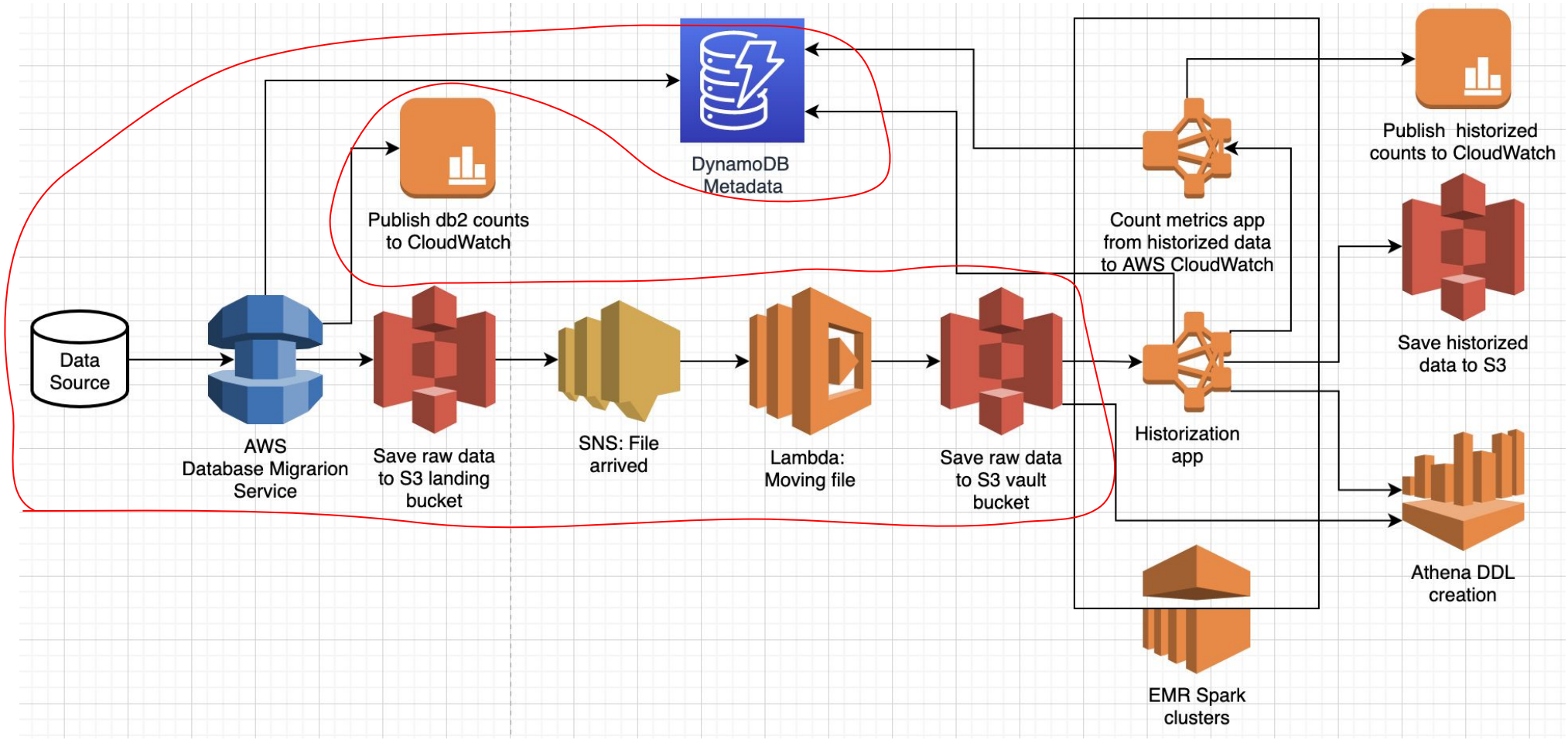
config.json

```
{
  "SchemaName": "schema_name",
  "TableName": "table_name",
  "Source": "db2",
  "Target": "s3",
  "StartValue": "1970-01-01 00:00:00.000000",
  "EndValue": "2021-05-11 00:00:00.000000",
  "PrimaryKeyFields": "pk_field",
  "TimestampColumnName": "",
  "OutputFormat": "parquet",
  "BrokenDateTimeColumns": "",
  "DecimalWithNullColumns": "",
  "IsAthenaTableRequired": true,
  "IsTargetPartitioned": false
}
```

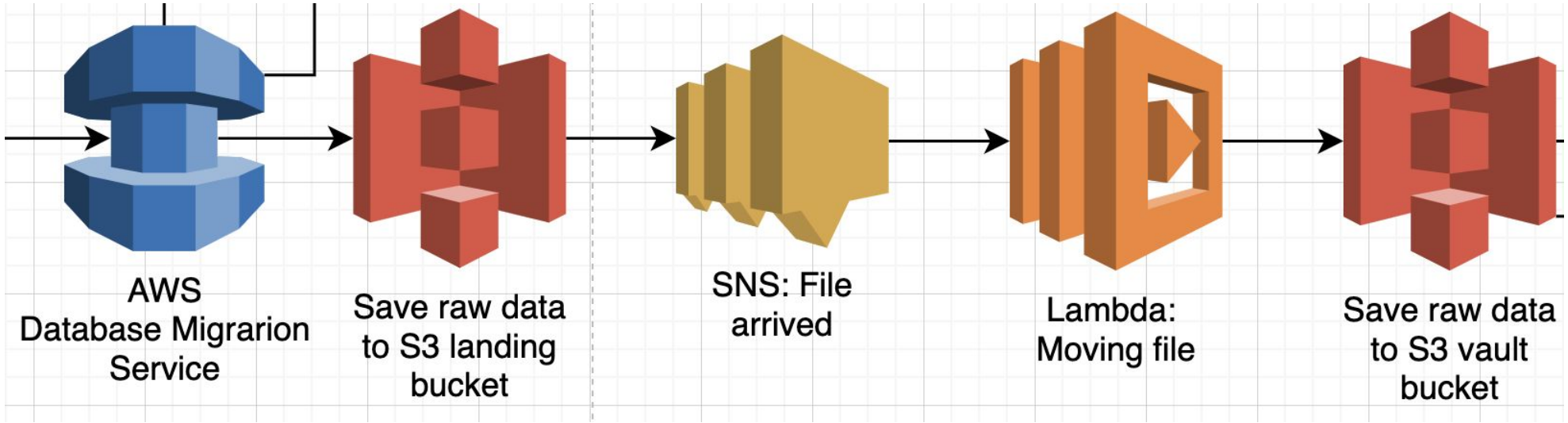
First part: Metadata management



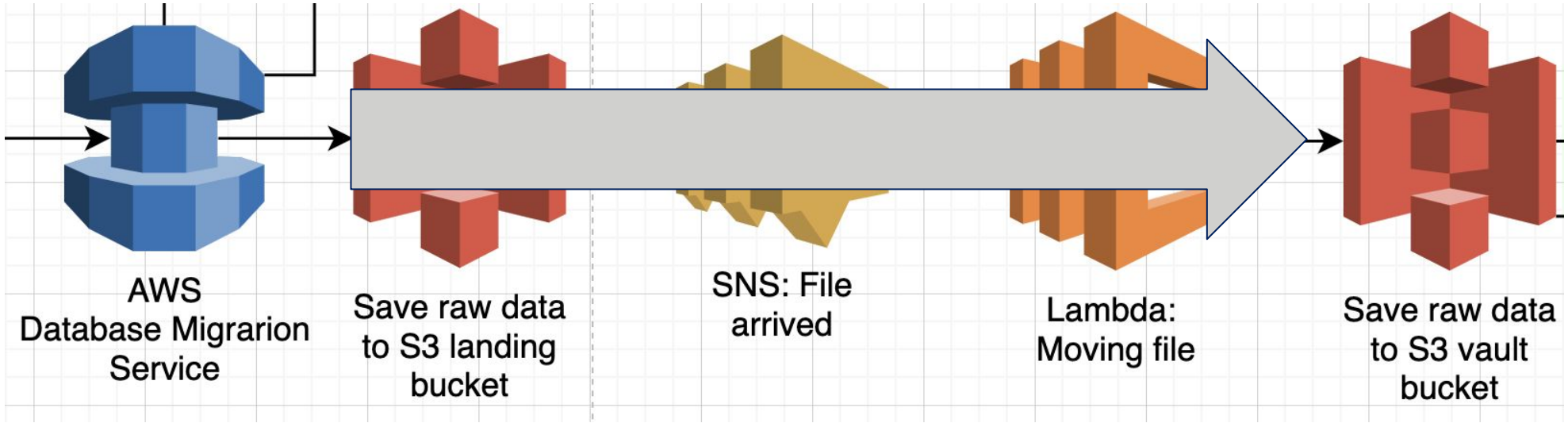
First part: DMS saving data



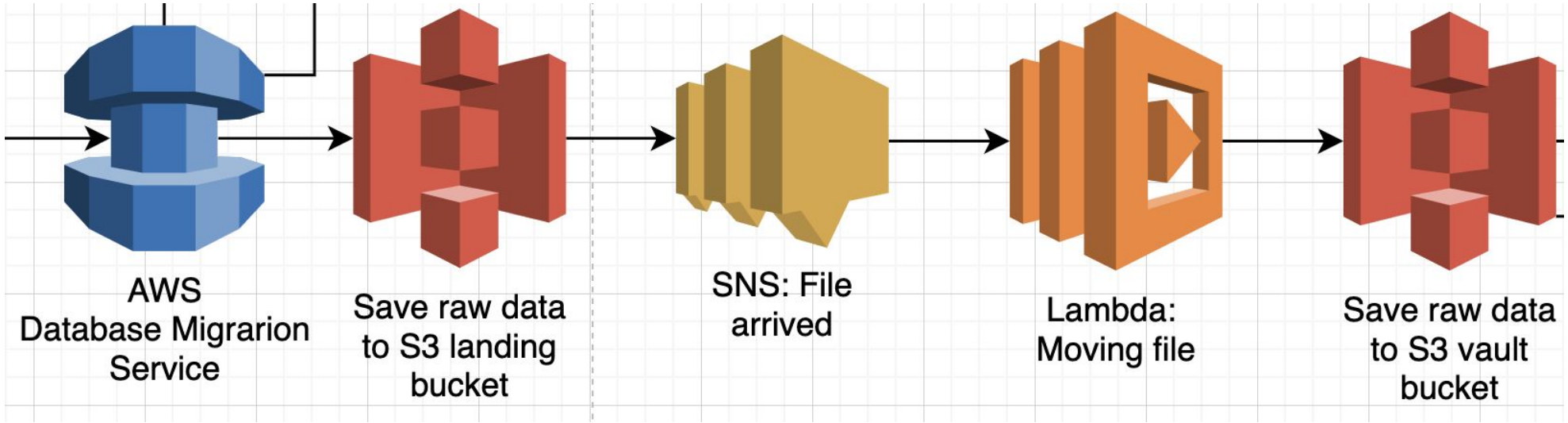
DMS: saving data

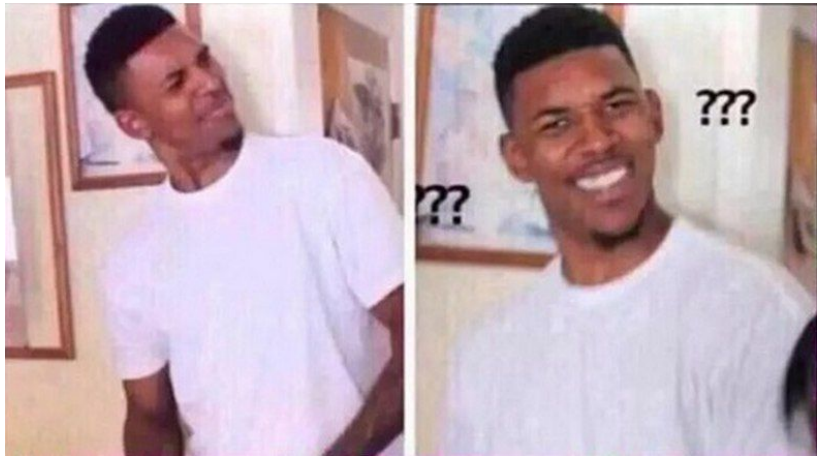


DMS: saving data

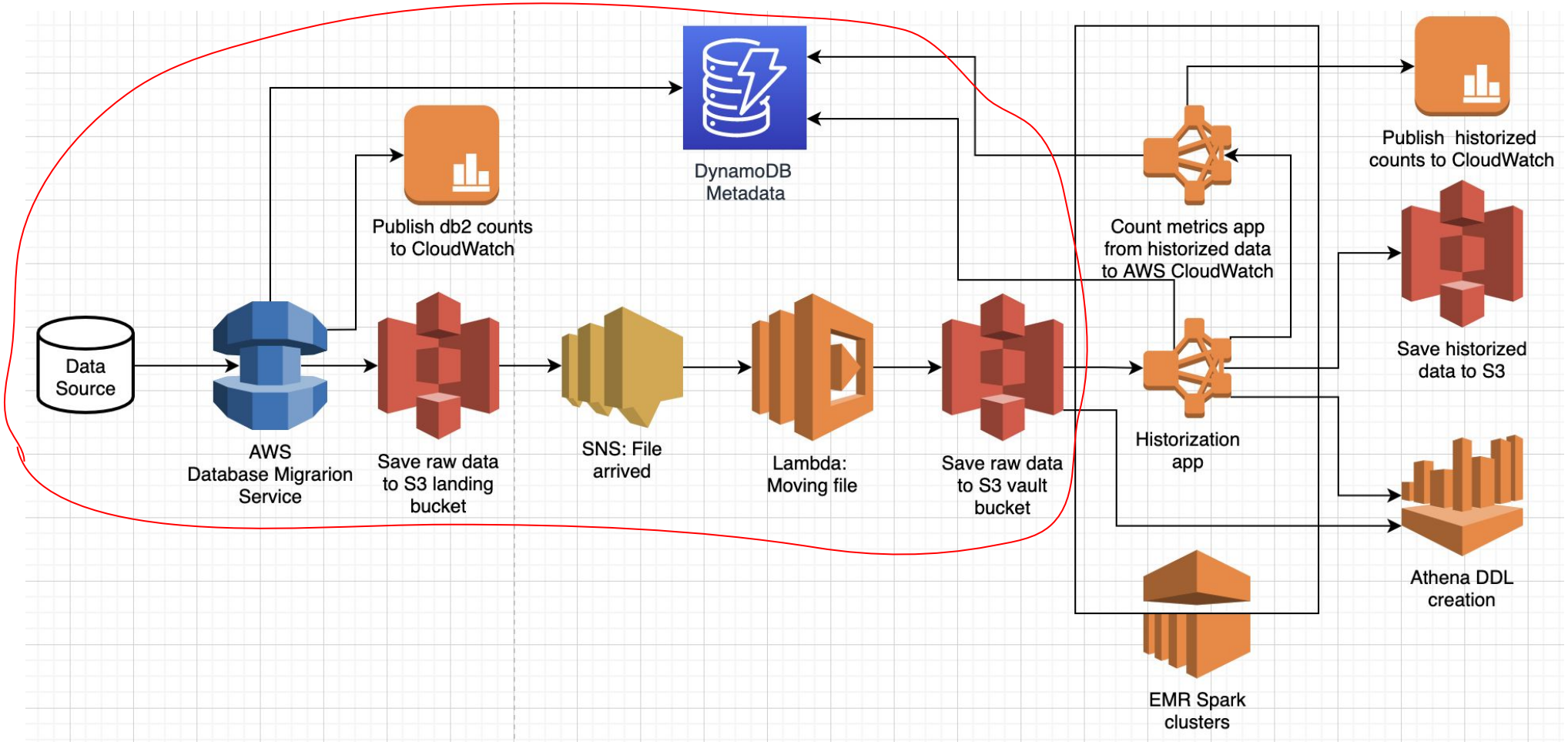


DMS: saving data

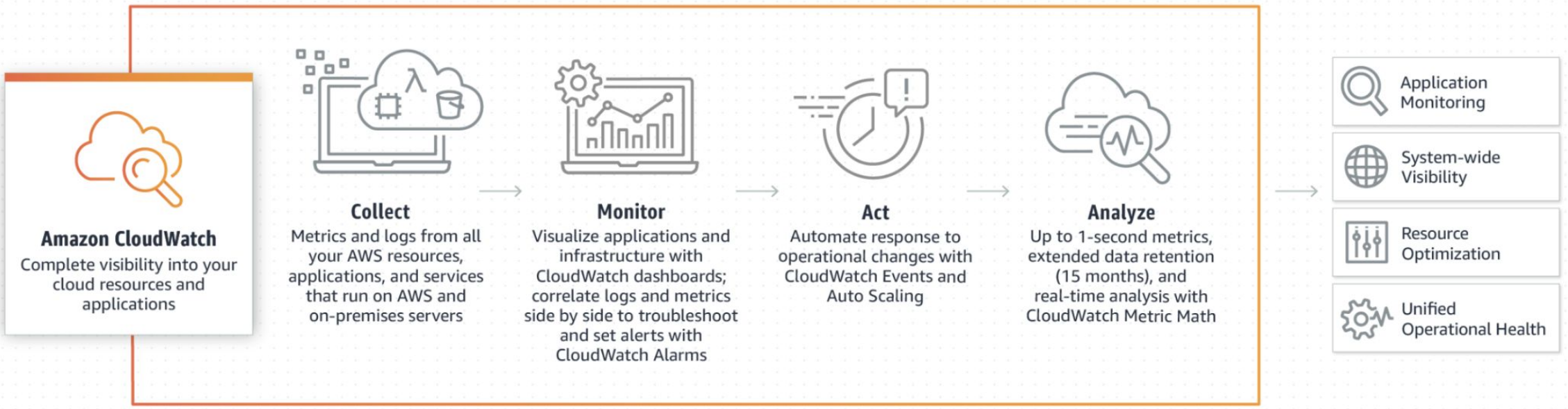




First part: DMS



AWS CloudWatch



AWS CloudWatch

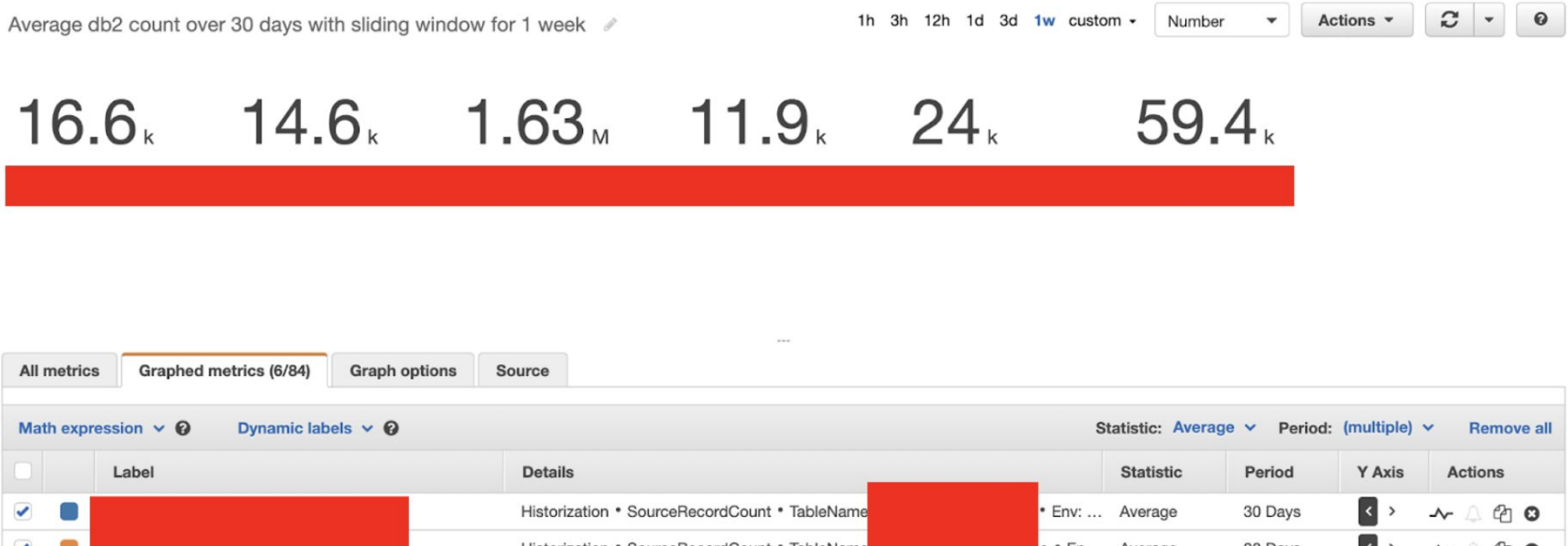
Data Platform product owner

AWS CloudWatch

Data Platform product owner

He is happy when he observes metrics

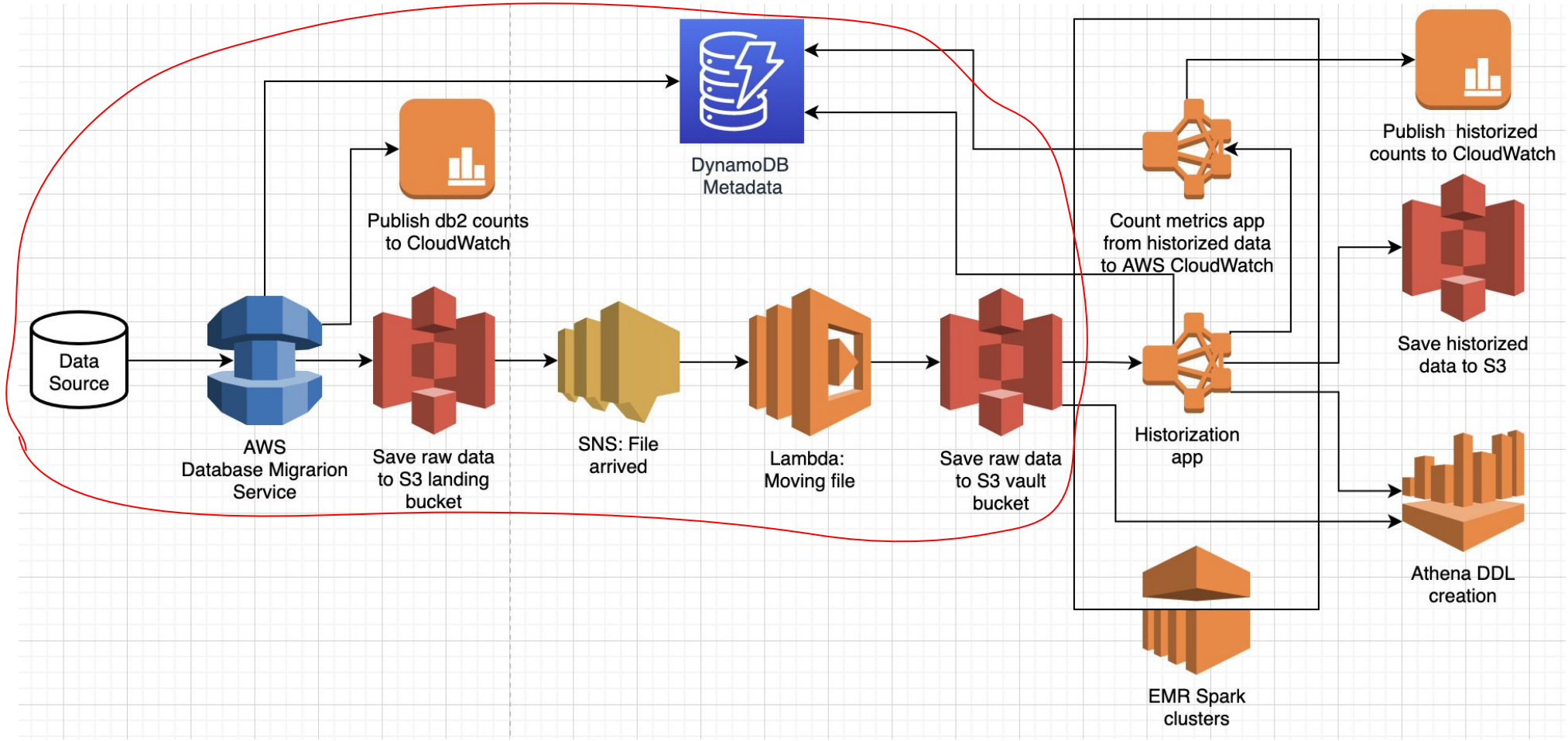
AWS CloudWatch



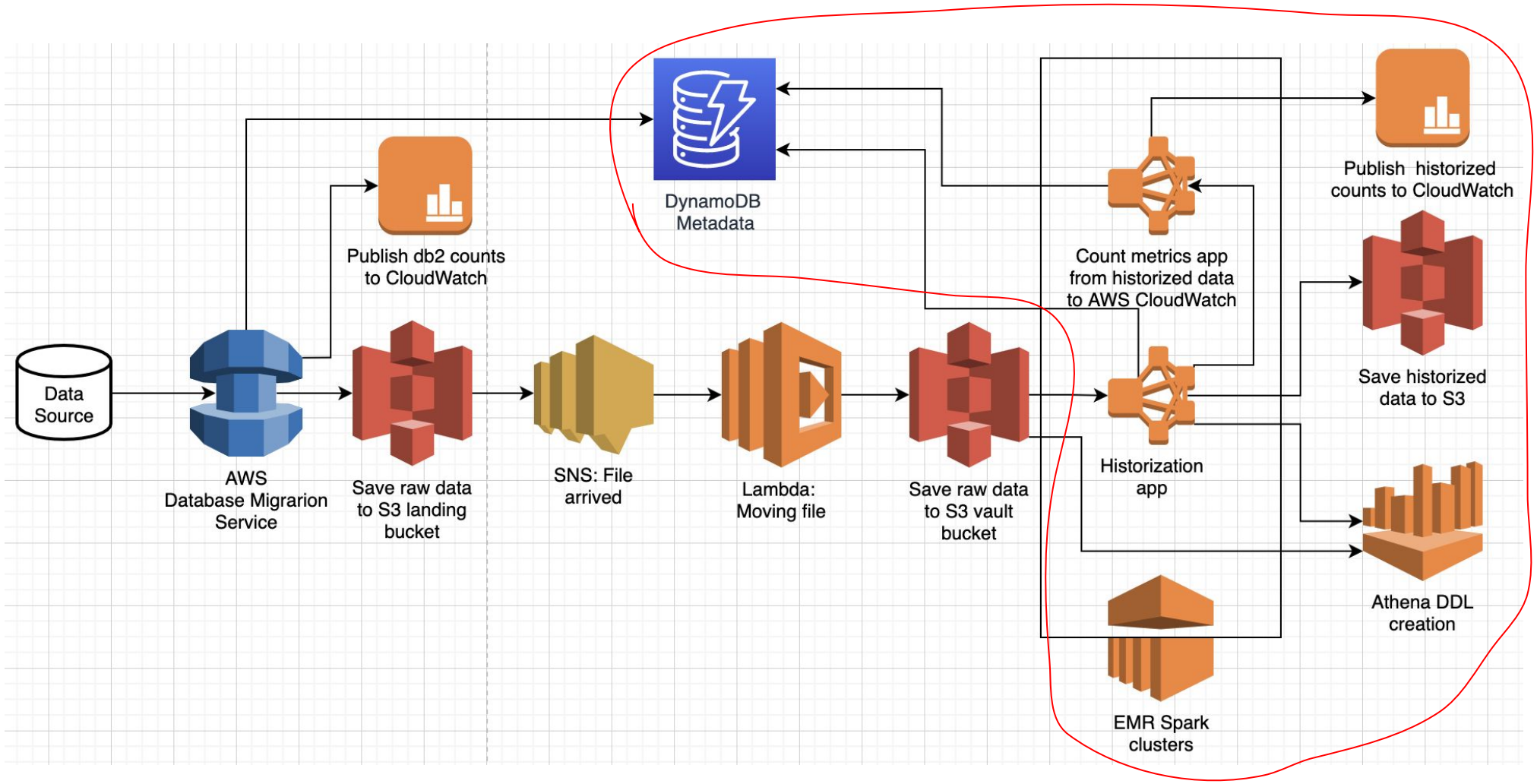
Data Platform product owner

He is happy when he observes metrics

First part: DMS



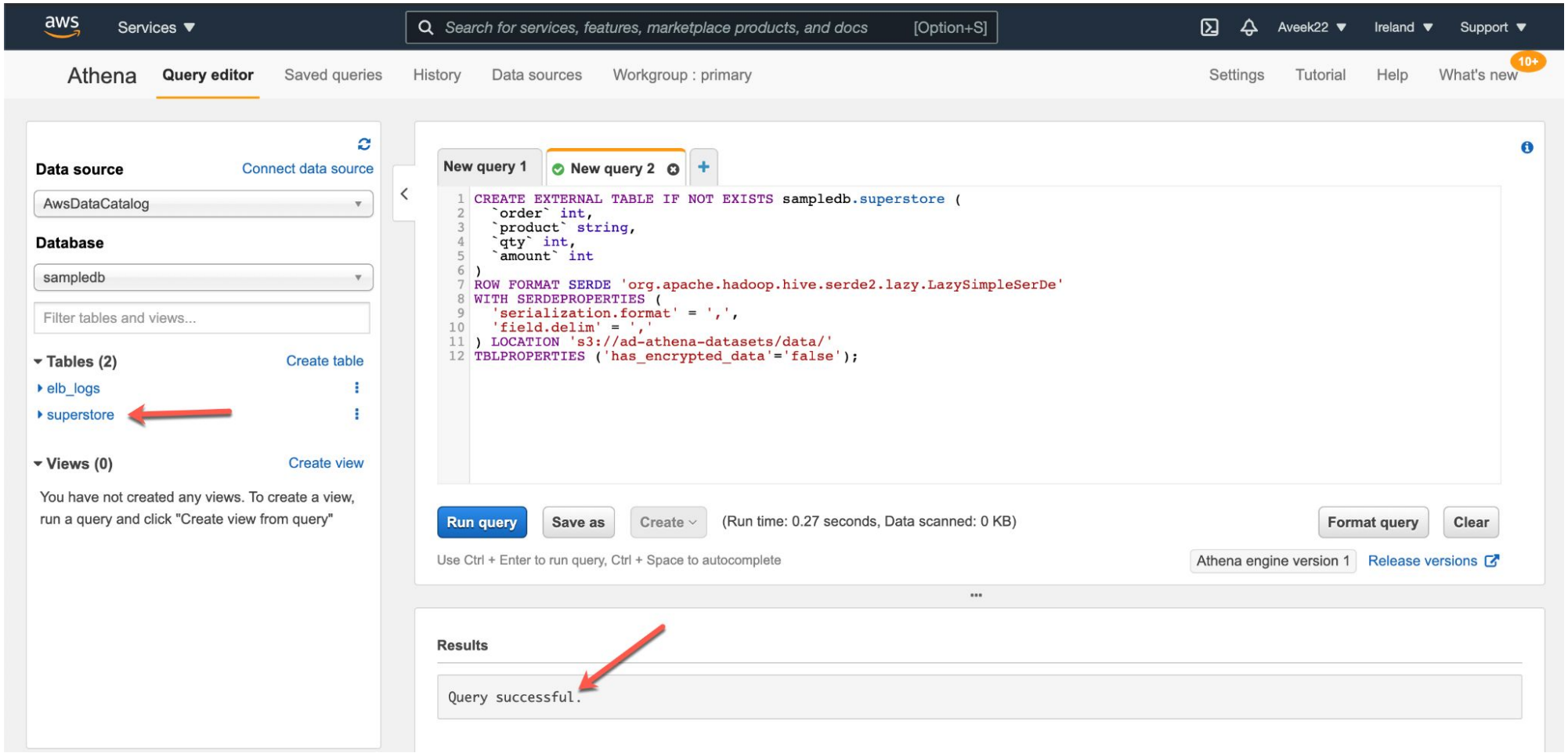
Second part: Historization



Historization: components

Historization: components

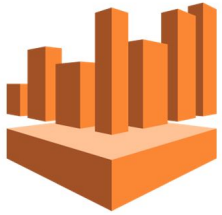
Historization: Athena



Historization: components

amazon
EMR

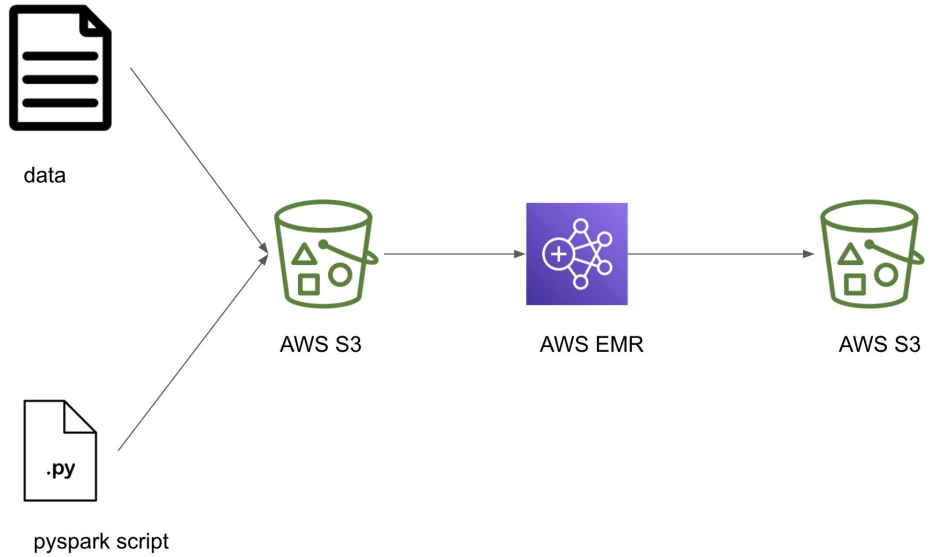
APACHE
Spark™



Amazon Athena

Spark jobs on EMR cluster

How it looks like in general

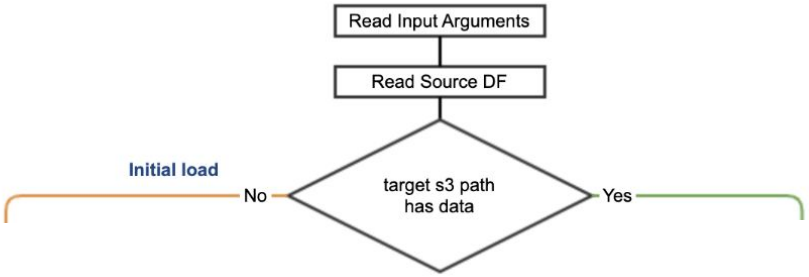


Historization: script input params

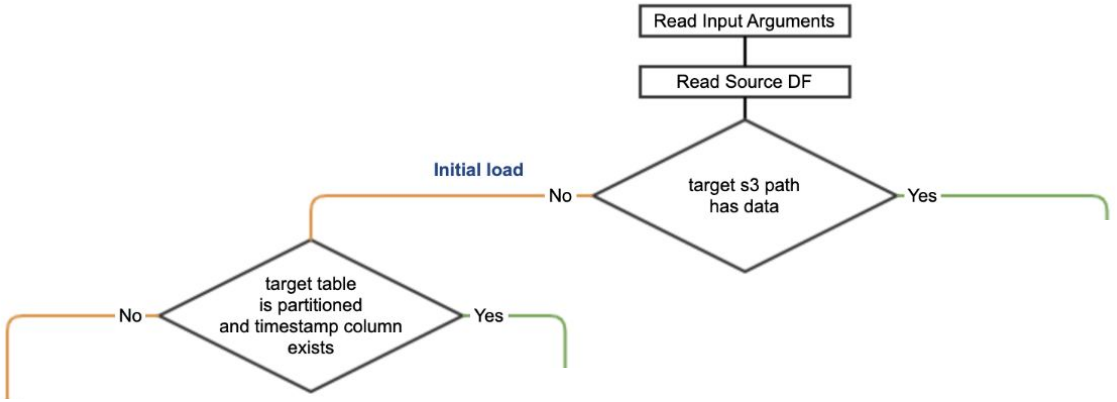
Input parameters

Name	Required	Type	Description
--source_s3_path	True	String	Source path for DMS outputs
--target_s3_path	True	String	Target path to the Delta Table
--load_date	True	String	Load date for source data in format YYYY-MM-DD
--primary_key_fields	True	String	List of fields (comma-separated string) used as primary key for source table
--timestamp_column_name	True	String	Timestamp column which will be used for historization to define valid_from and valid_to fields, default 'CTS'
--is_target_partitioned	False	Bool	True if target table is partitioned, default is 'True'
--is_athena_table_required	False	Bool	True if you want to create/update Athena table and you're sure that table doesn't contain PII data, default is 'False'

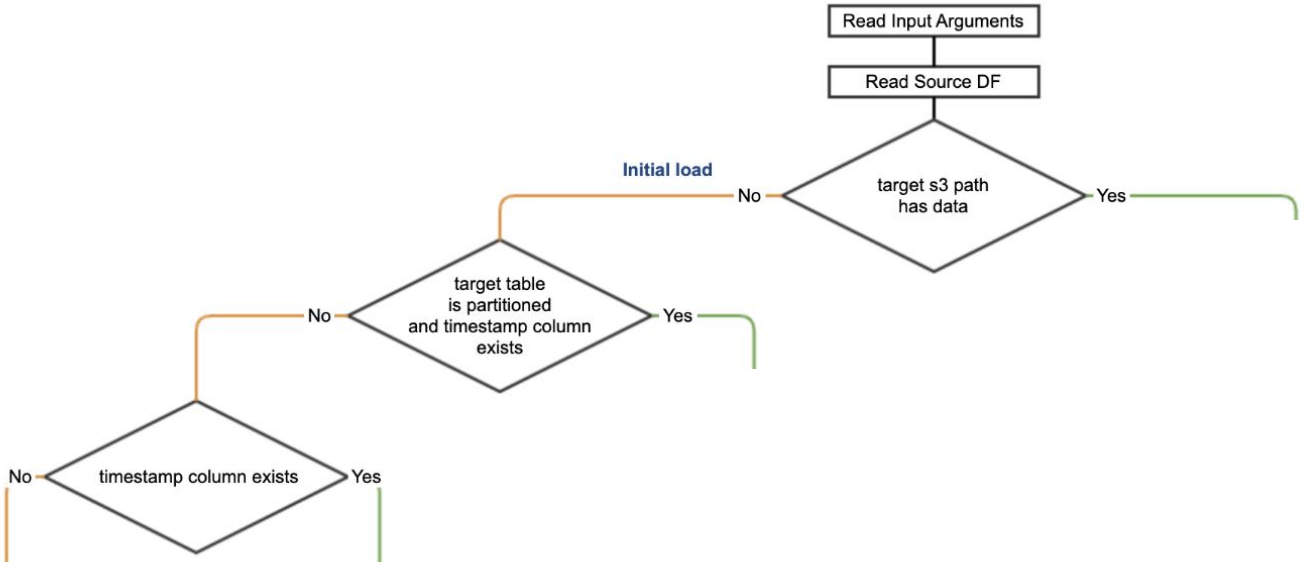
Historization: algorithm



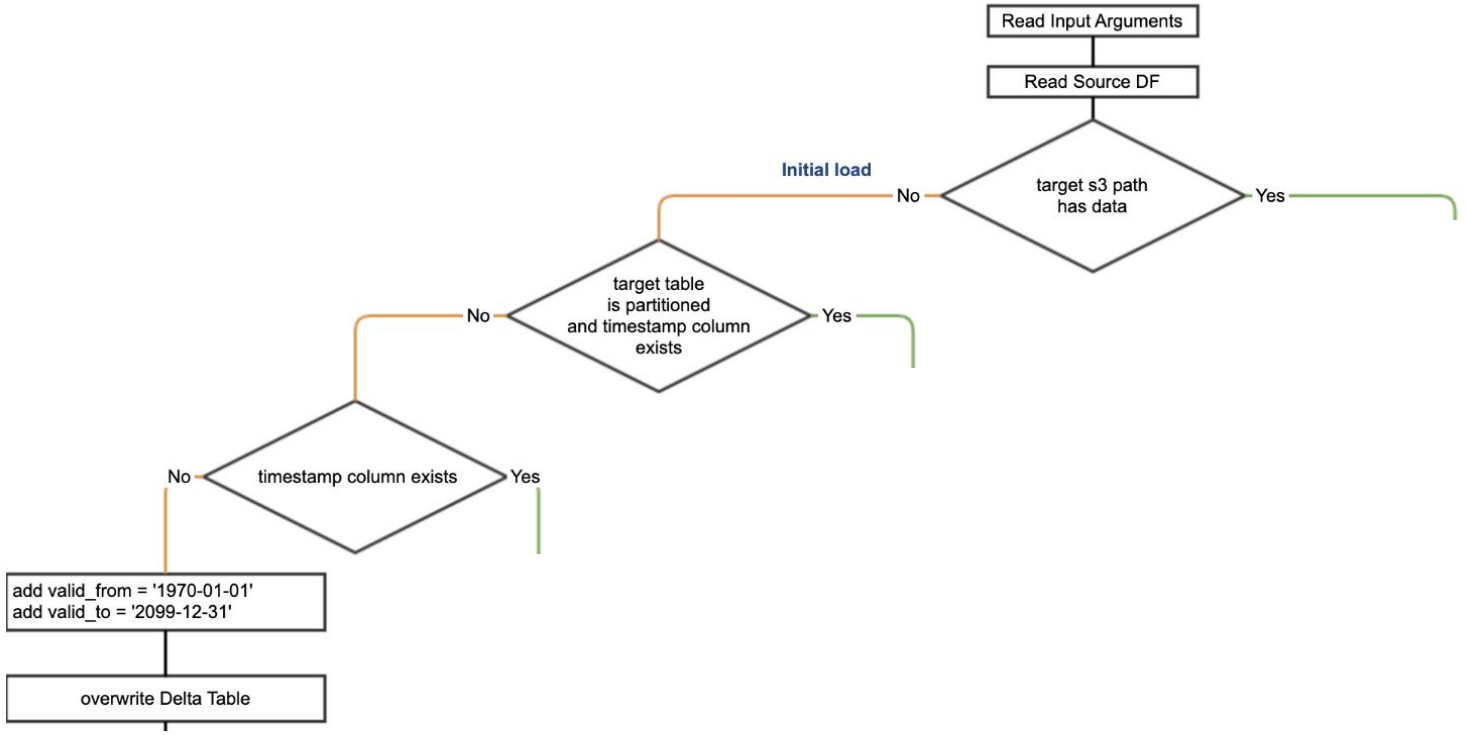
Historization: algorithm



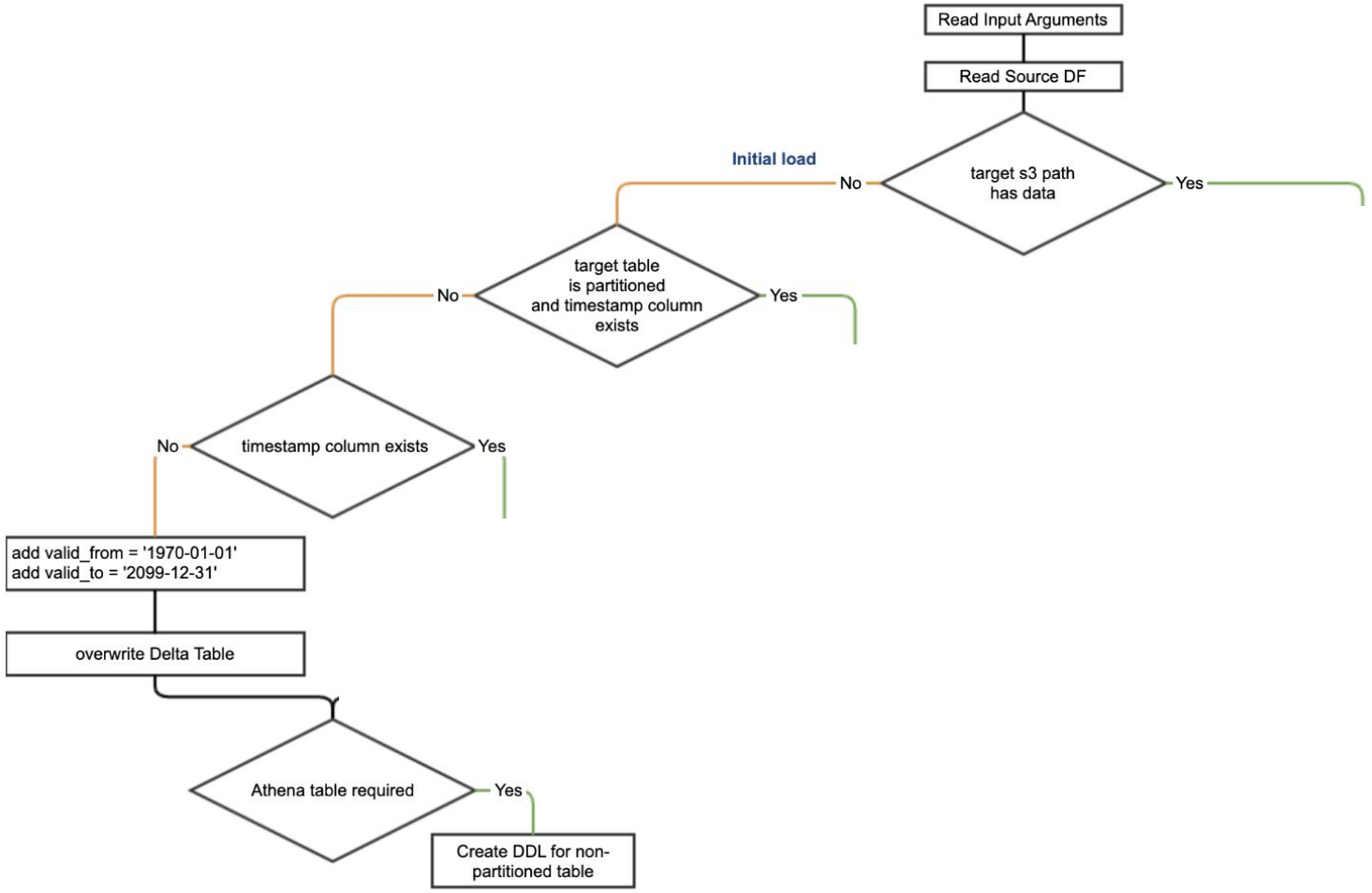
Historization: algorithm



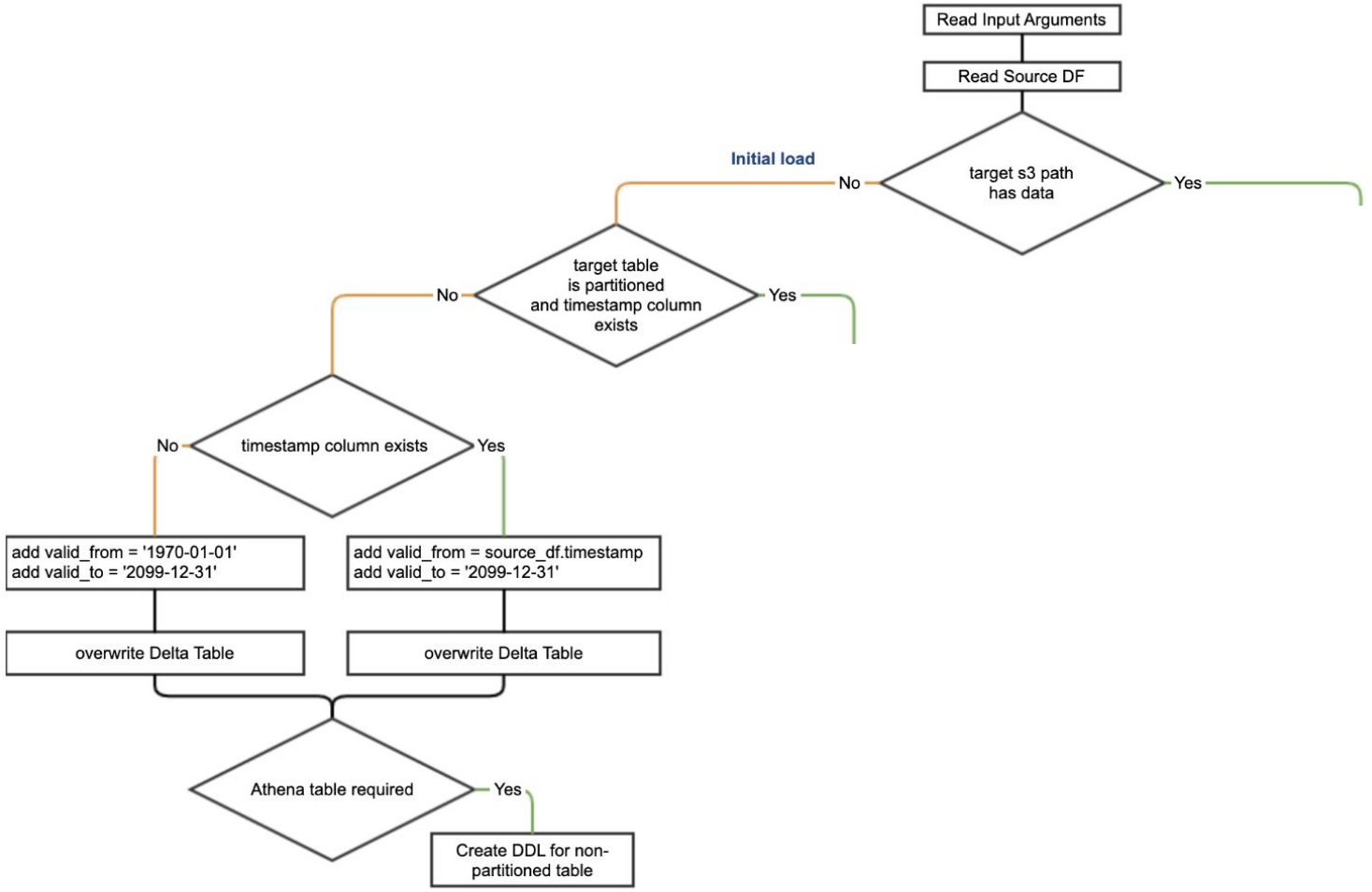
Historization: algorithm



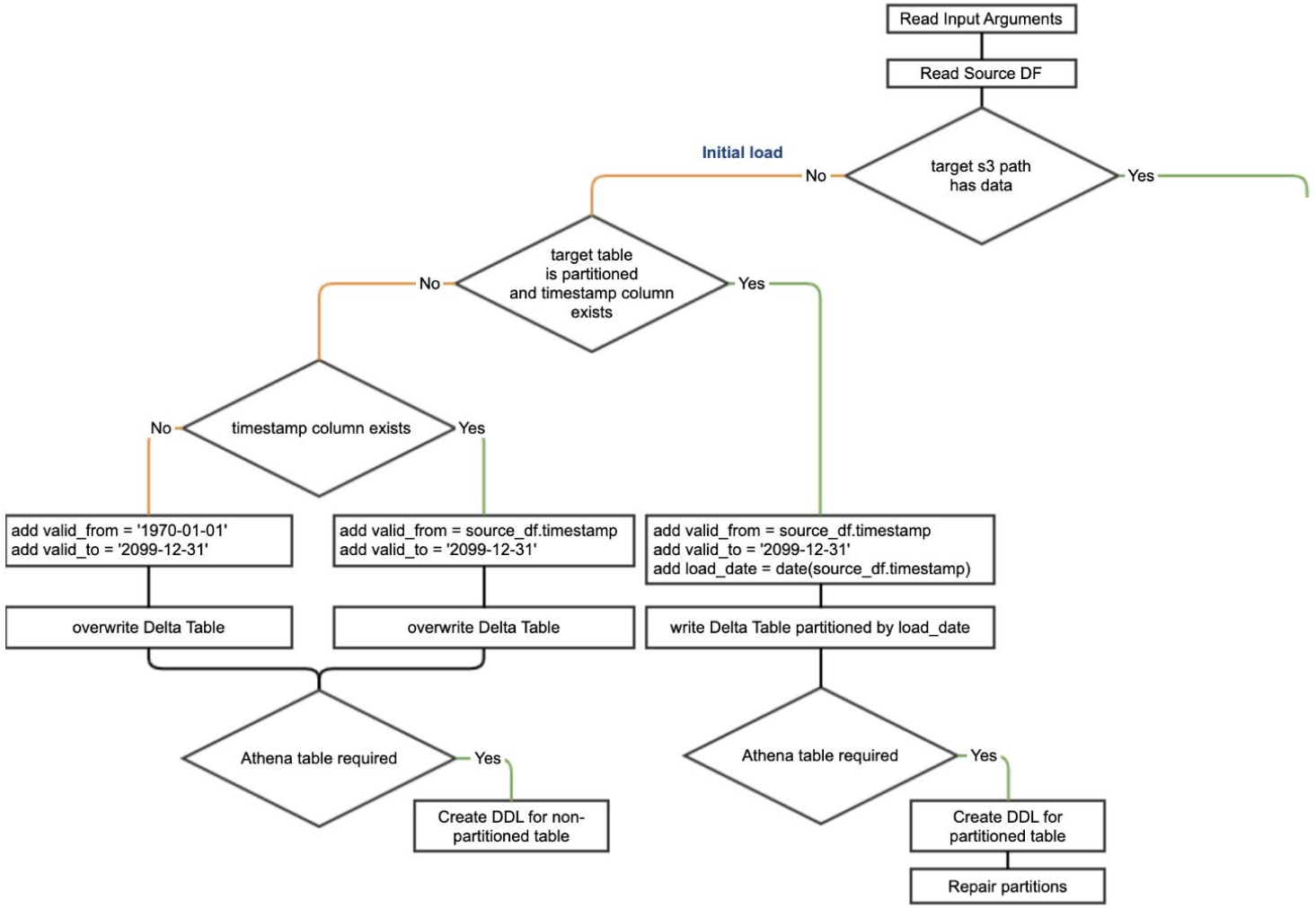
Historization: algorithm



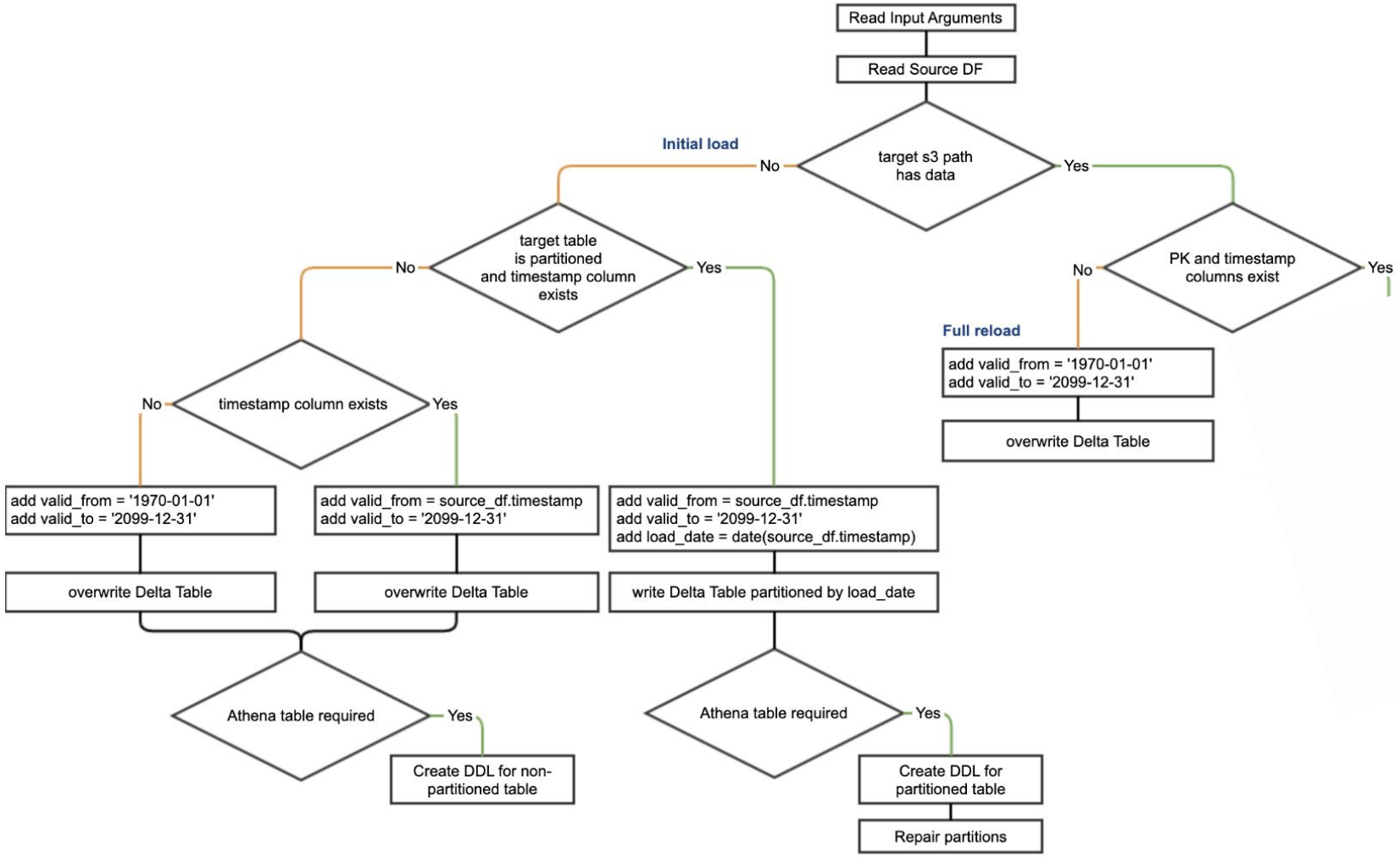
Historization: algorithm



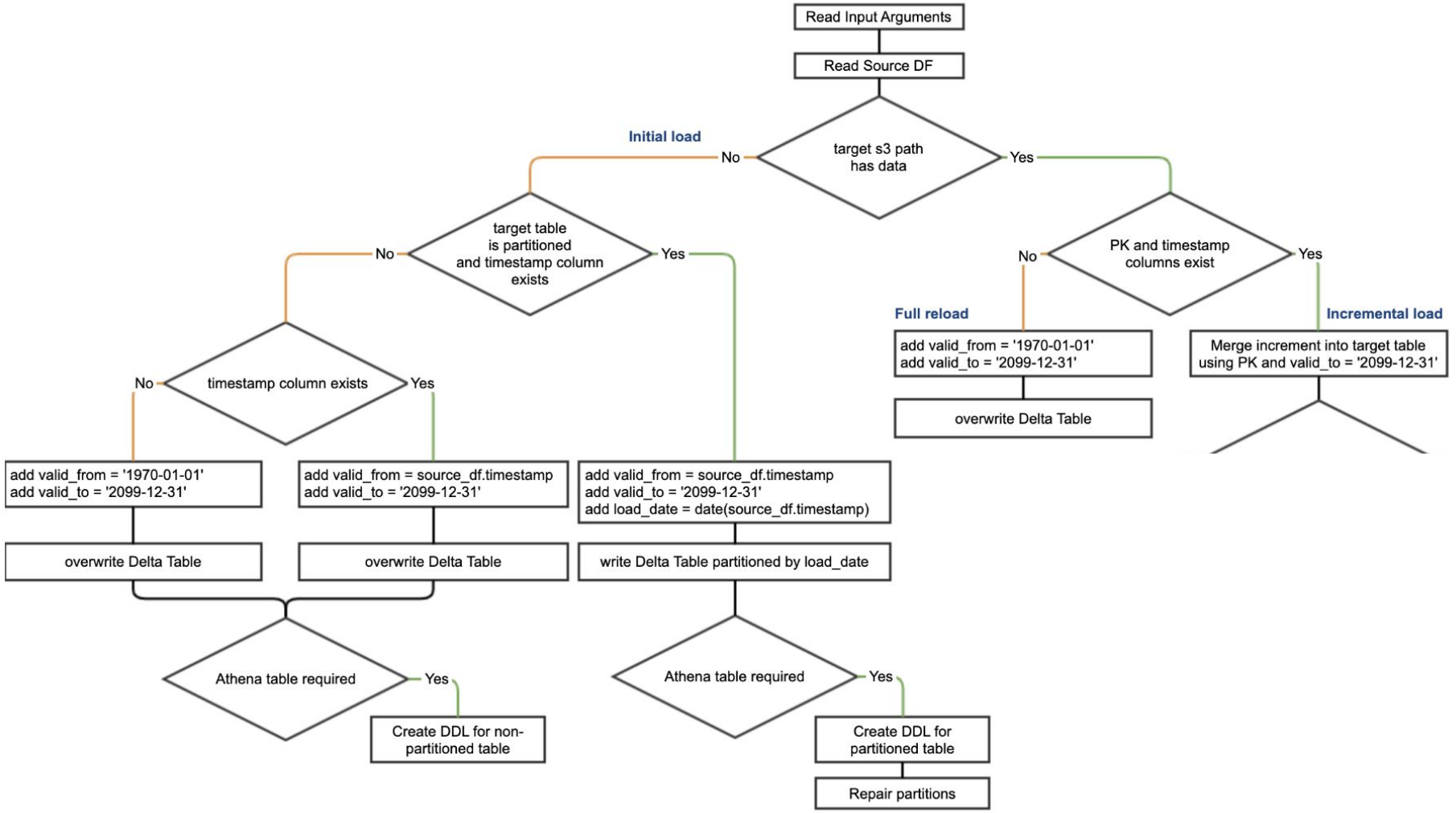
Historization: algorithm



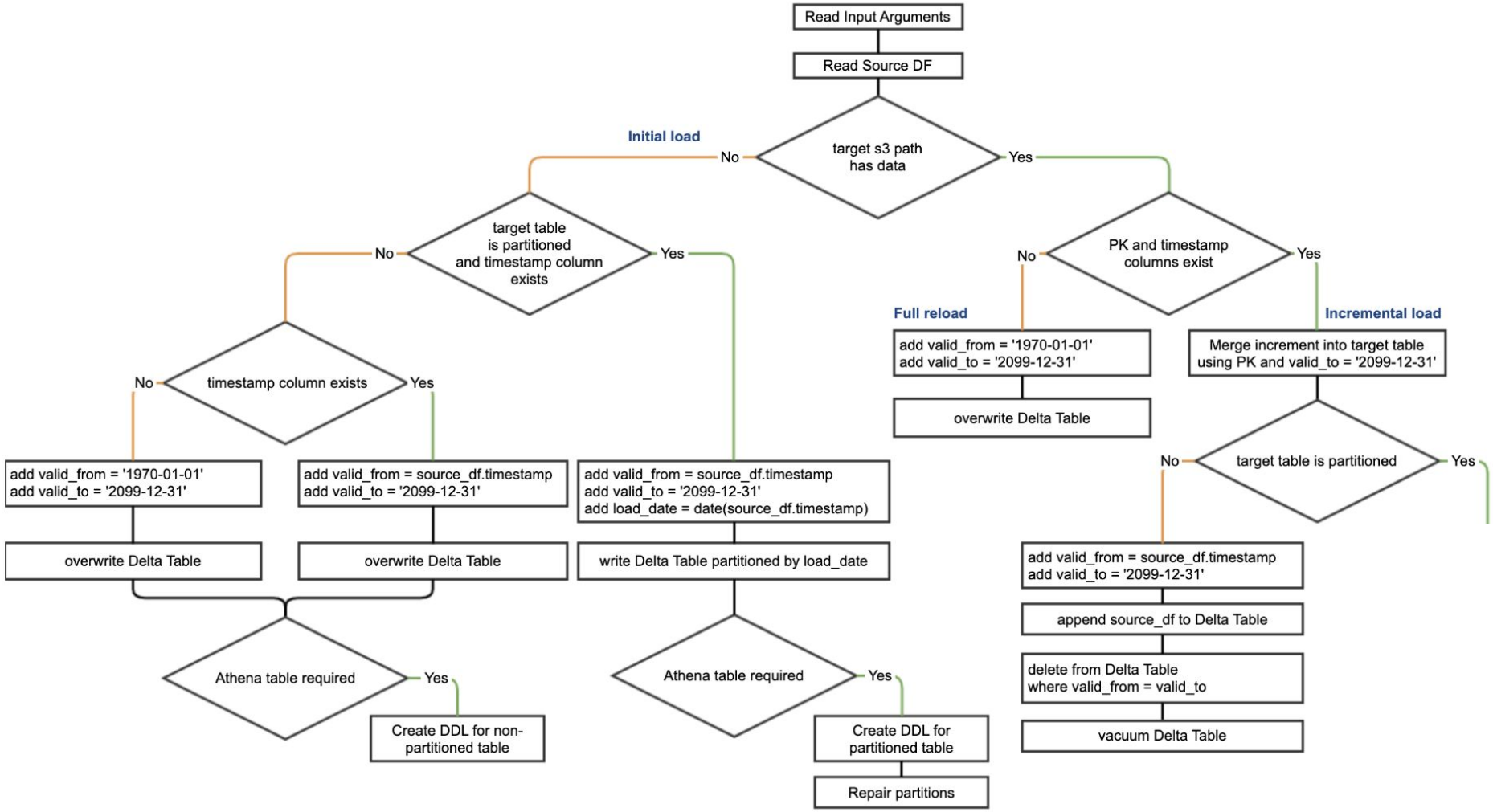
Historization: algorithm



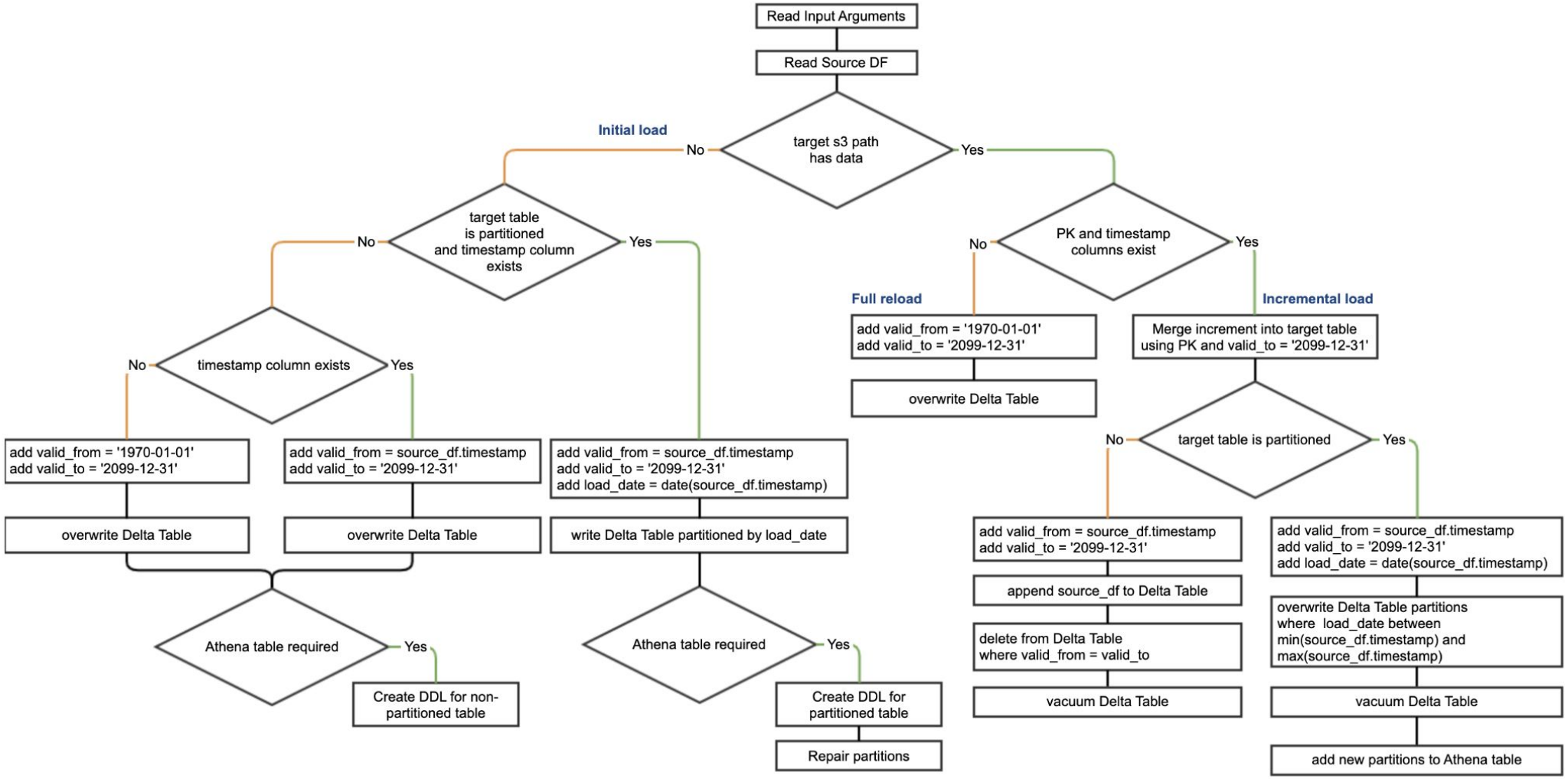
Historization: algorithm



Historization: algorithm



Historization: algorithm

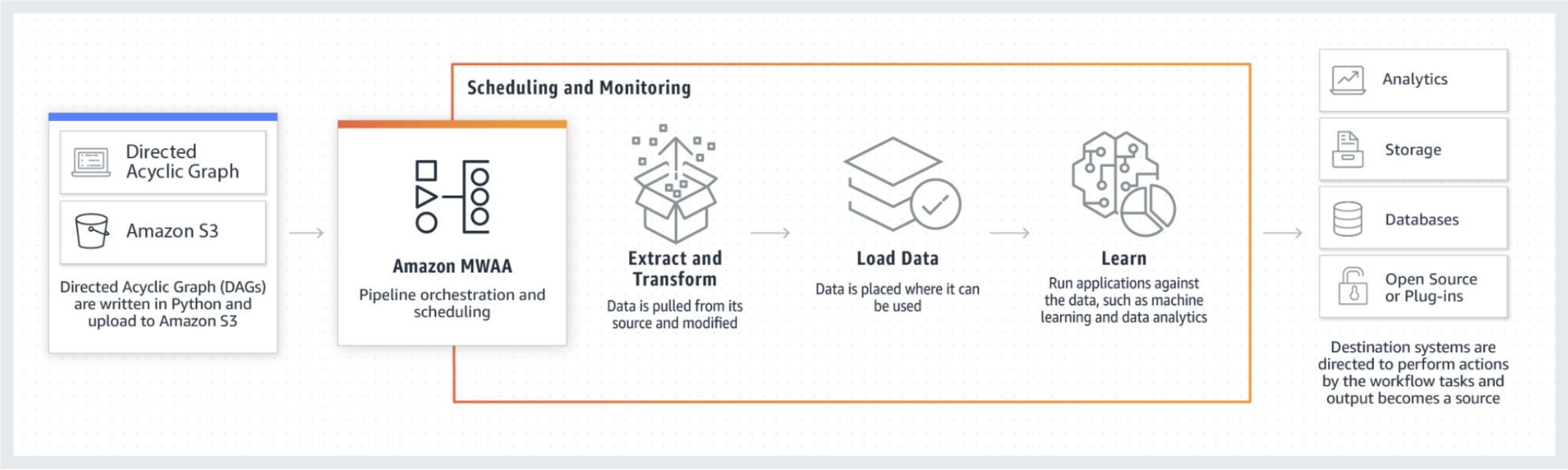


Workflow management platform

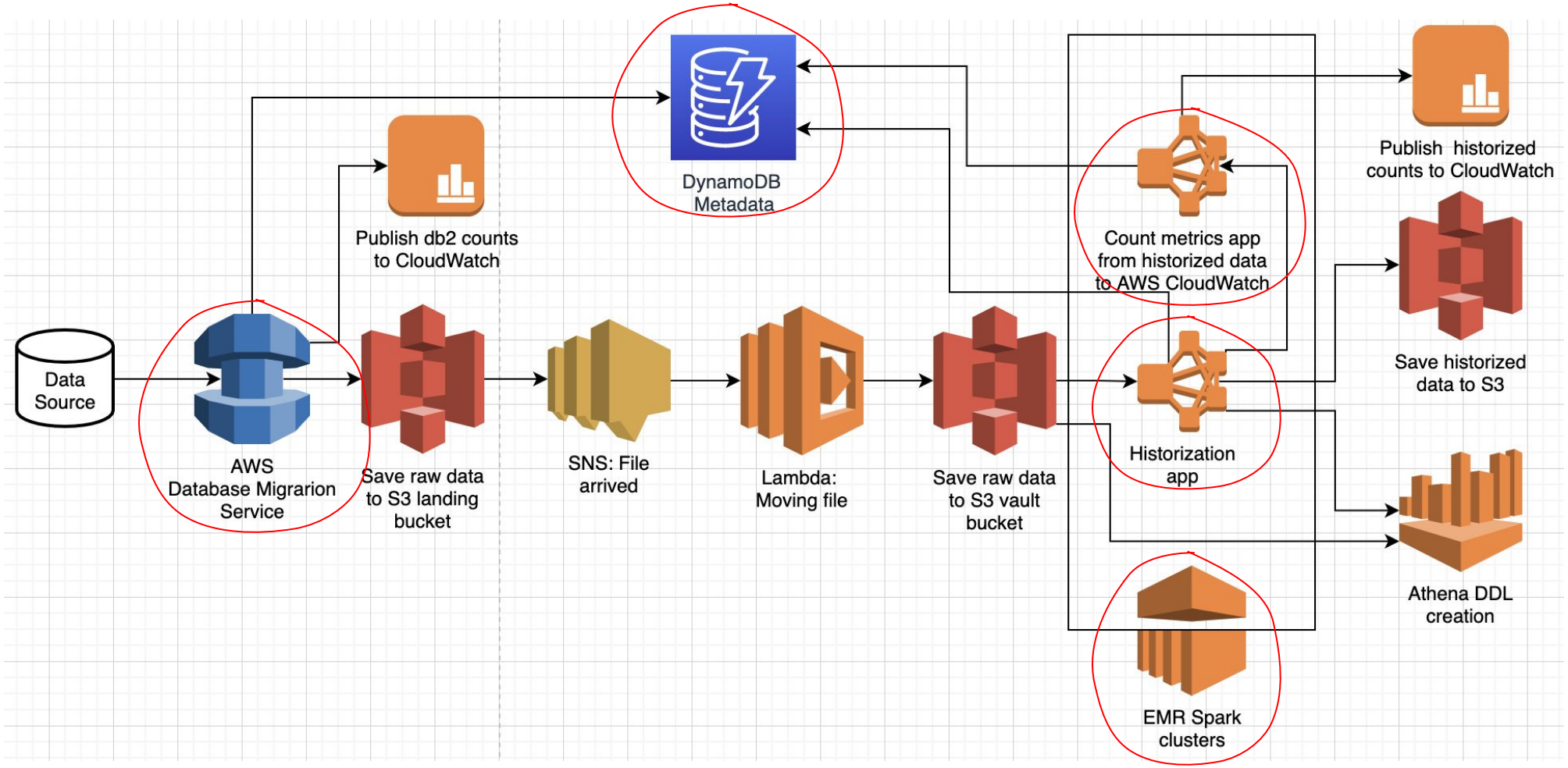
The workflow is orchestrated by MWAA

Workflow management platform

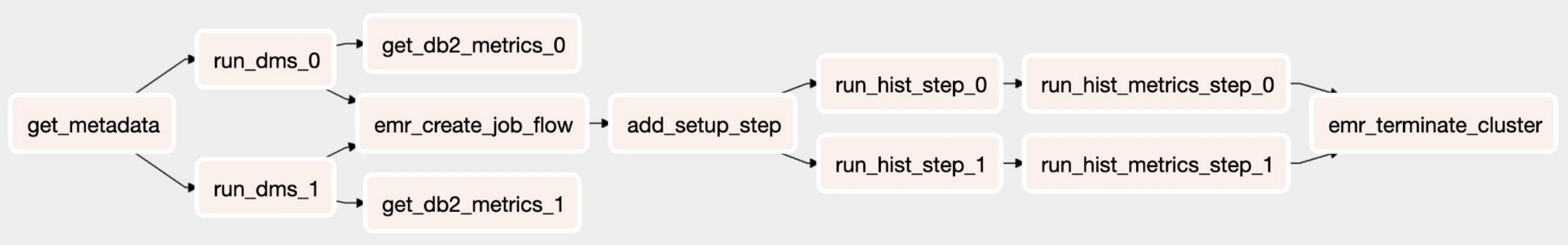
The workflow is orchestrated by MWAA



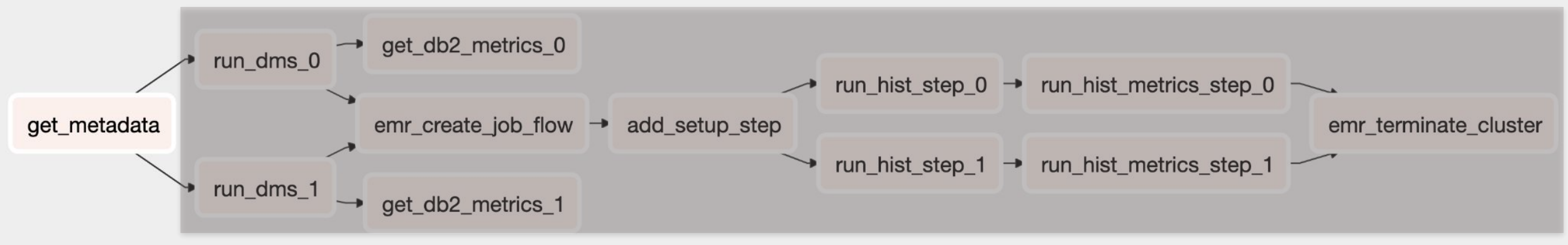
Airflow: Dag first iteration



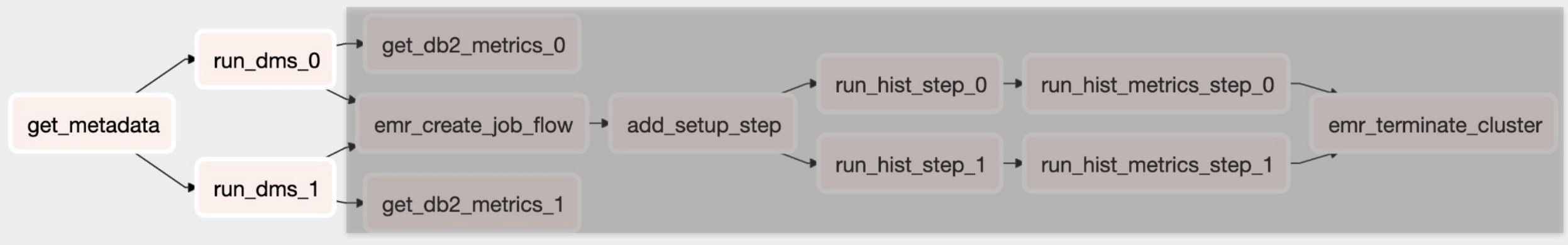
Airflow: Dag first iteration



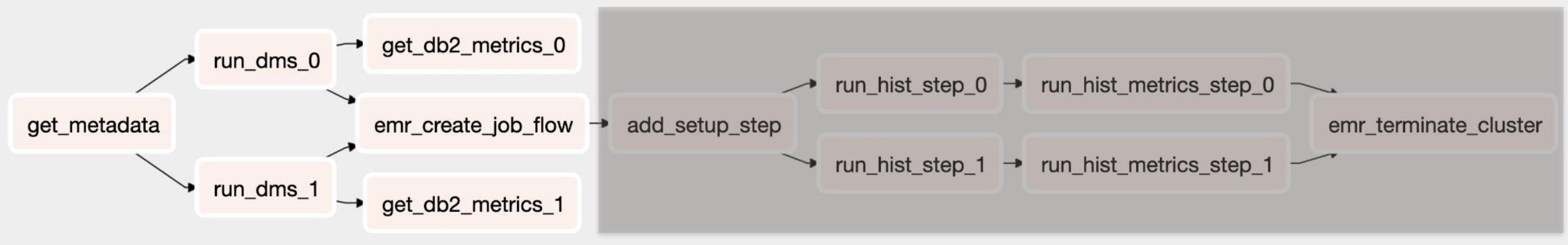
Airflow: Dag first iteration



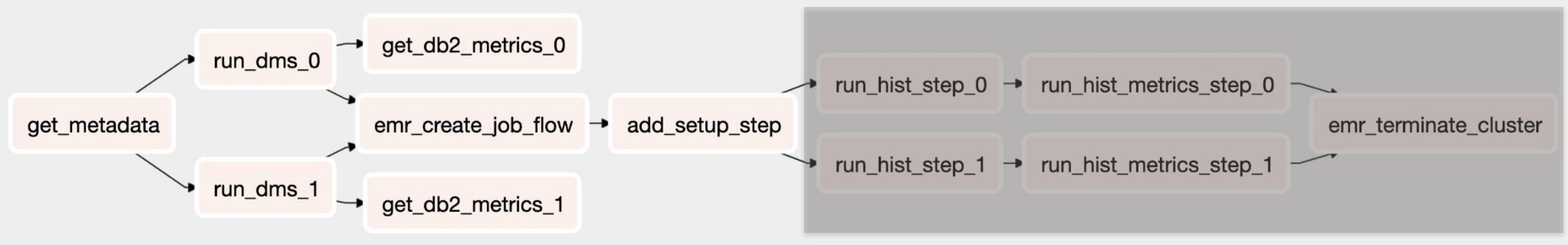
Airflow: Dag first iteration



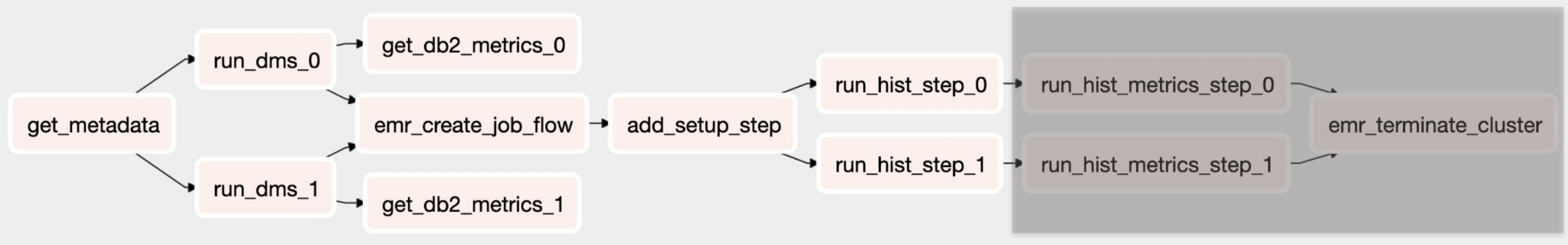
Airflow: Dag first iteration



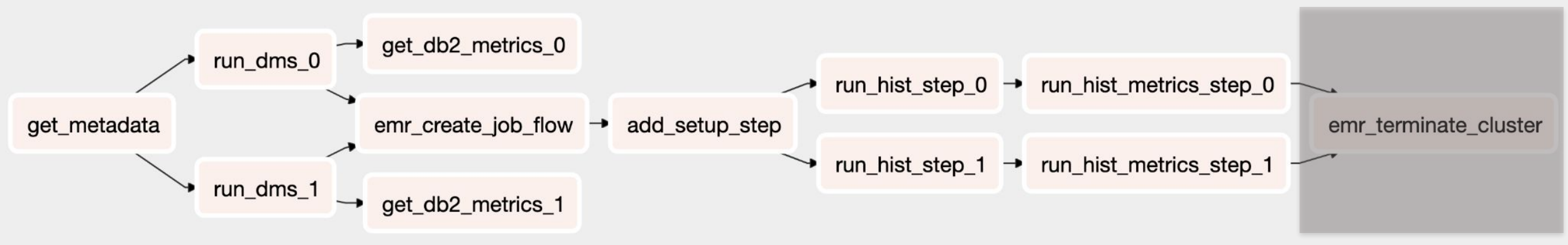
Airflow: Dag first iteration



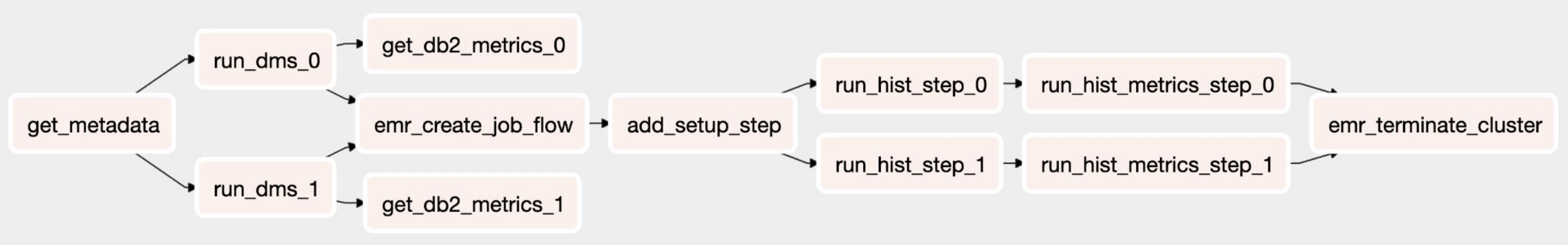
Airflow: Dag first iteration



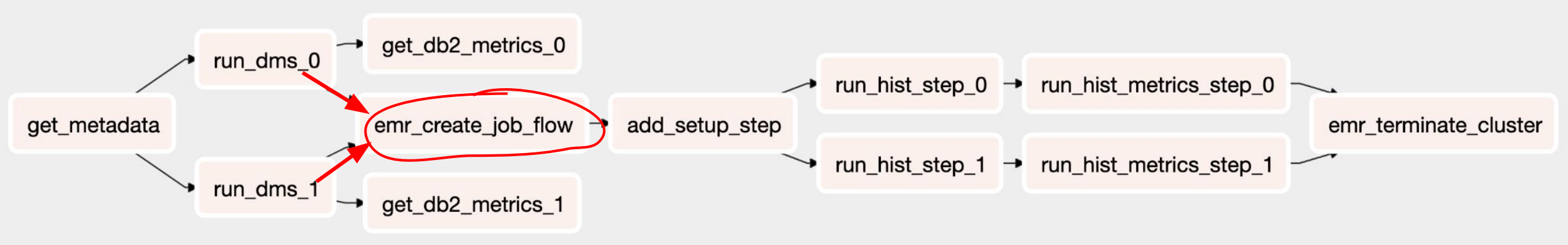
Airflow: Dag first iteration



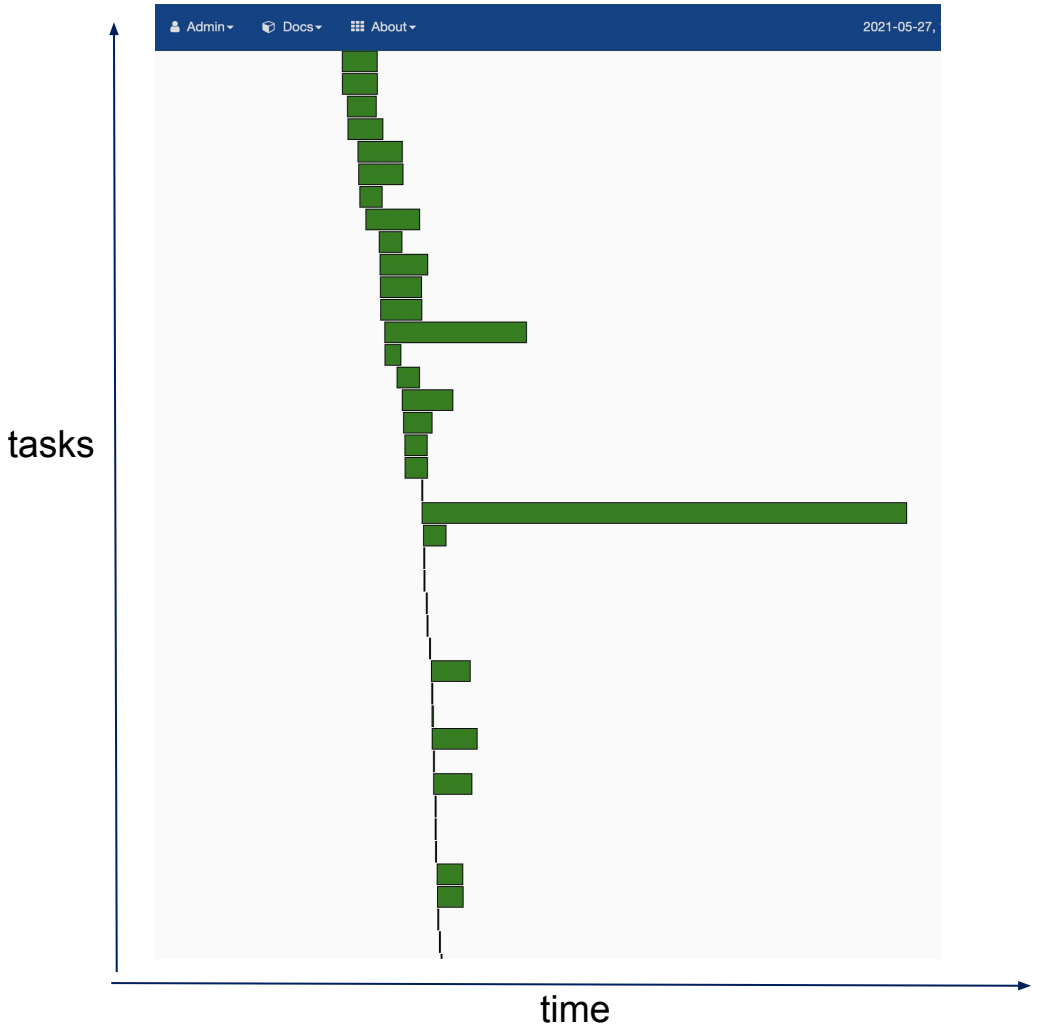
Airflow: Dag first iteration



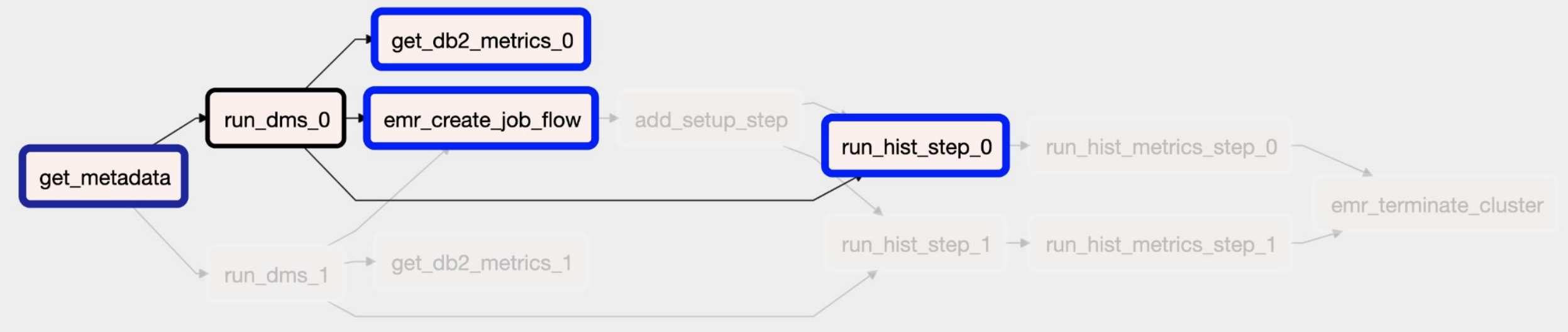
Airflow: Dag first iteration



Dag first iteration: Gantt diagramm



Airflow: Dag second iteration



Airflow: Monitoring and alerting

Airflow: Monitoring

```
# Airflow DAG Config
DEFAULT_ARGS = {
    'email': ['airflow@example.com'],
    'email_on_failure': True,
    'email_on_retry': False,
    'on_failure_callback': send_zoom_failure_message,
    'retries': 3,
    'retry_delay': timedelta(minutes=3)
}
```

Airflow: Monitoring

```
# Airflow DAG Config
DEFAULT_ARGS = {
    'email': ['airflow@example.com'],
    'email_on_failure': True,
    'email_on_retry': False,
    'on_failure_callback': send_zoom_failure_message,
    'retries': 3,
    'retry_delay': timedelta(minutes=3)
}
```

Airflow: Monitoring

```
# Airflow DAG Config
DEFAULT_ARGS = {
    'email': ['airflow@example.com'],
    'email_on_failure': True,
    'email_on_retry': False,
    'on_failure_callback': send_zoom_failure_message,
    'retries': 3,
    'retry_delay': timedelta(minutes=3)
}
```

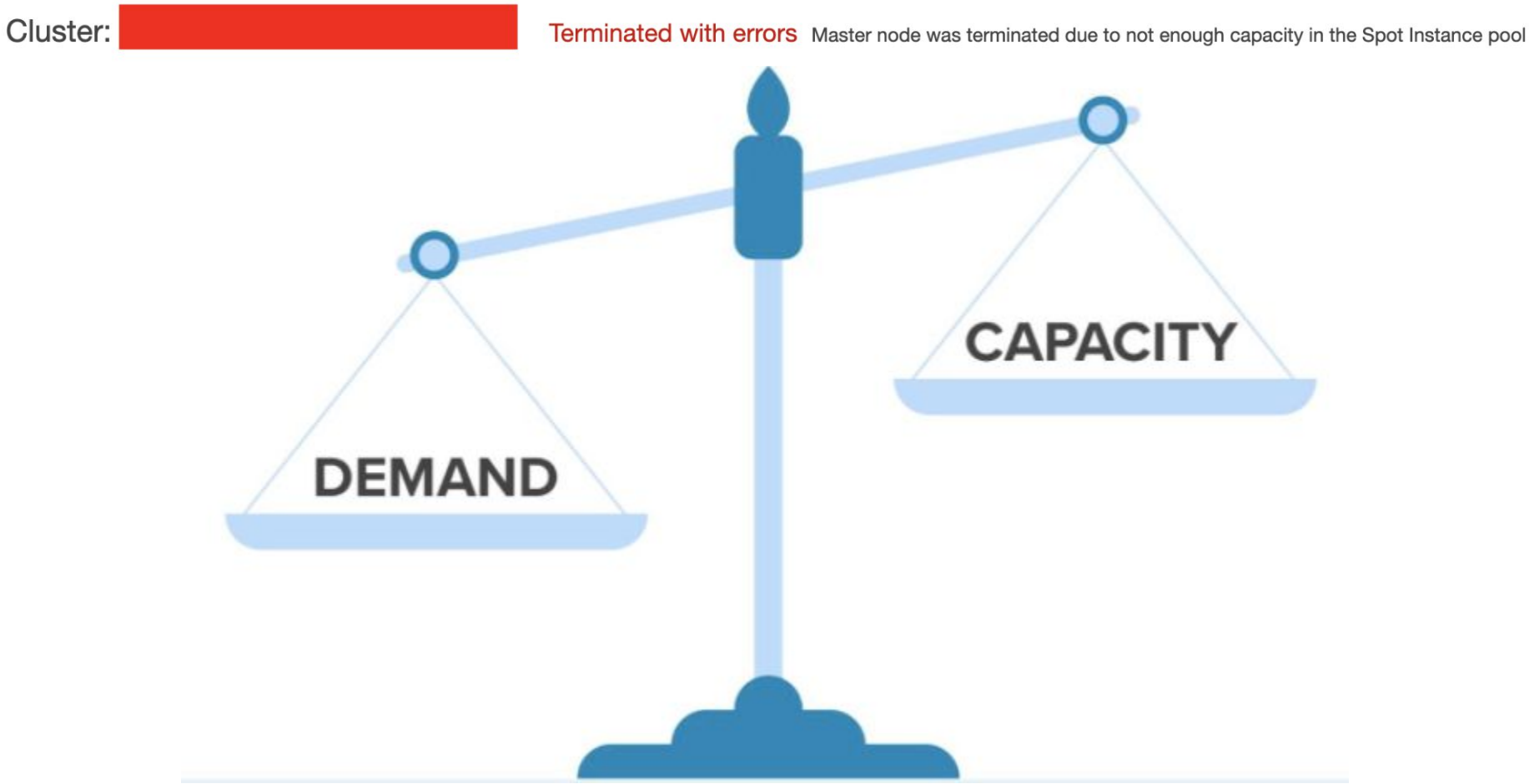
Airflow: points to improve

EMR EC2 Spot Instances

Name your own price for EC2 Compute

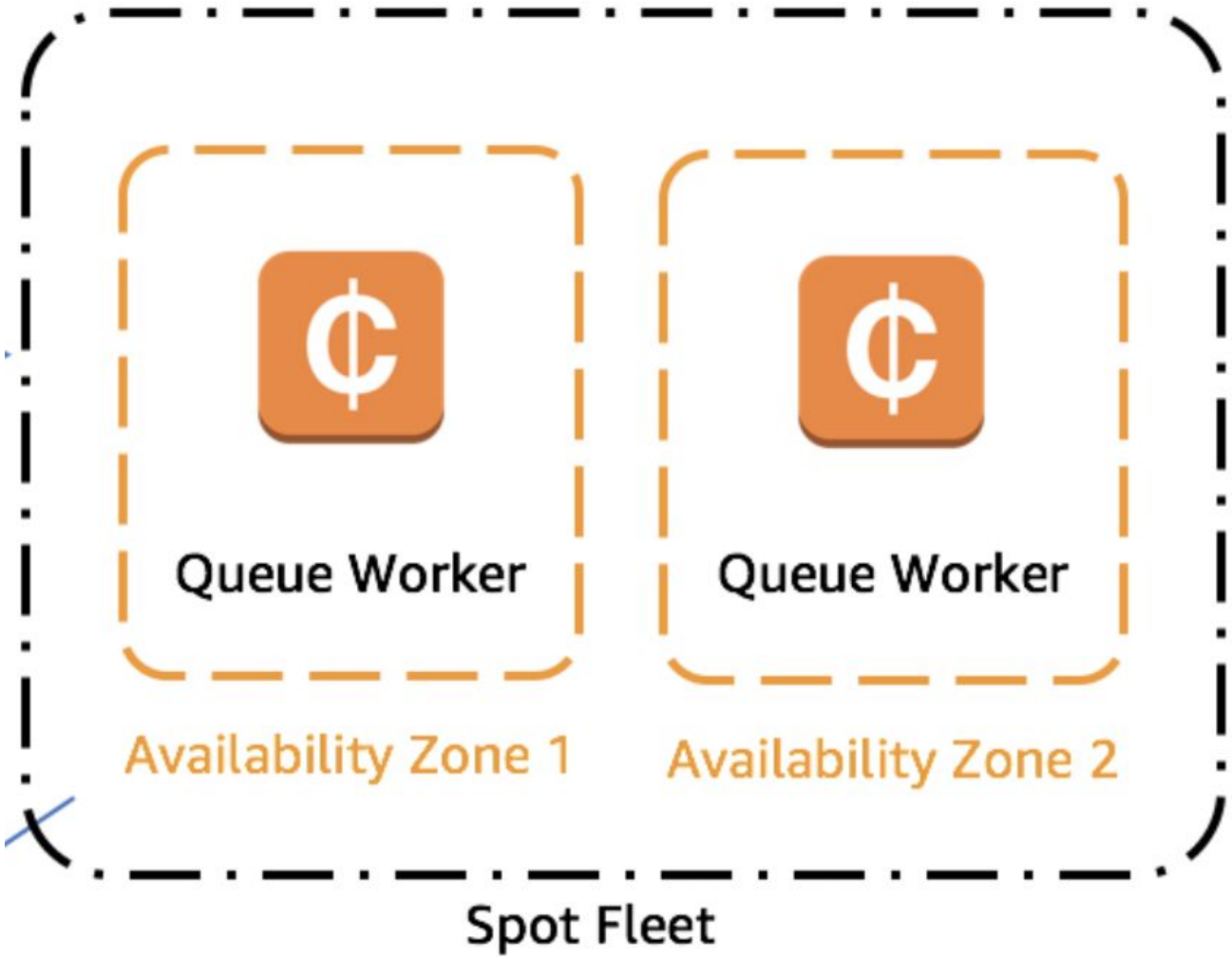
- A market where price of compute changes based upon Supply and Demand
- When Bid Price exceeds Spot Market Price, instance is launched
- Instance is terminated (with 2 minute warning) if market price exceeds bid price
- Unused On-Demand Instances

Airflow: points to improve



Airflow: ec2 fleet

- The instance fleets configuration for a cluster offers the widest variety of provisioning options for EC2 instances
- With instance fleets, you specify target capacities for On-Demand Instances and Spot Instances within each fleet.



That’s it?

That’s it?

Data type issues

- time_millis

Data type issues

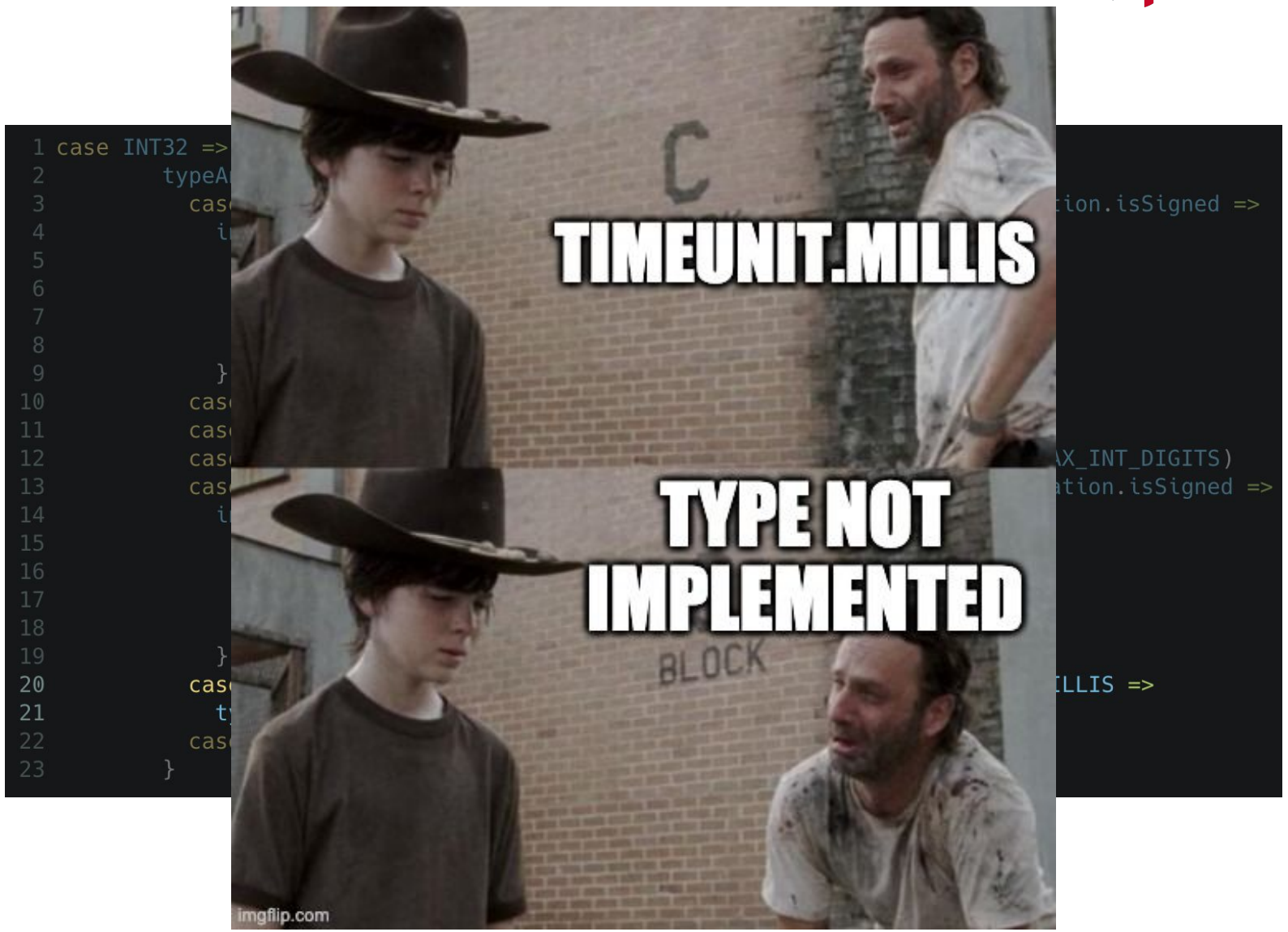
- time_millis

Spark ParquetSchemaConverter.scala

```
case INT32 =>
        typeAnnotation match {
          case intTypeAnnotation: IntLogicalTypeAnnotation if intTypeAnnotation.isSigned =>
            intTypeAnnotation.getBitWidth match {
              case 8 => ByteType
              case 16 => ShortType
              case 32 => IntegerType
              case _ => illegalType()
            }
          case null => IntegerType
          case _: DateLogicalTypeAnnotation => DateType
          case _: DecimalLogicalTypeAnnotation => makeDecimalType(Decimal.MAX_INT_DIGITS)
          case intTypeAnnotation: IntLogicalTypeAnnotation if !intTypeAnnotation.isSigned =>
            intTypeAnnotation.getBitWidth match {
              case 8 => ShortType
              case 16 => IntegerType
              case 32 => LongType
              case _ => illegalType()
            }
          case t: TimestampLogicalTypeAnnotation if t.getUnit == TimeUnit.MILLIS =>
            typeNotImplemented()
          case _ => illegalType()
        }
```

Data type issues

- time_millis



Data type issues

- time_millis
- inadequate dates

Data type issues

- time_millis
- inadequate dates

```
63
64  SELECT
65        min(                    ) AS min_date1,
66        max(                    ) AS max_date1
67    FROM
68
```

MIN_DATE1
0006-05-12

Data type issues

- time_millis
- inadequate dates

```
63
64  SELECT
65        min(                    ) AS min_date1,
66        max(                    ) AS max_date1
67    FROM
68
```

MIN_DATE1
0006-05-12

Data type issues

- time_millis
- inadequate dates

```
63
64  SELECT
65        min(                    ) AS min_date1,
66        max(                    ) AS max_date1
67    FROM
68
```

MIN_DATE1	MAX_DATE1
0006-05-12	2580-05-15

Data type issues

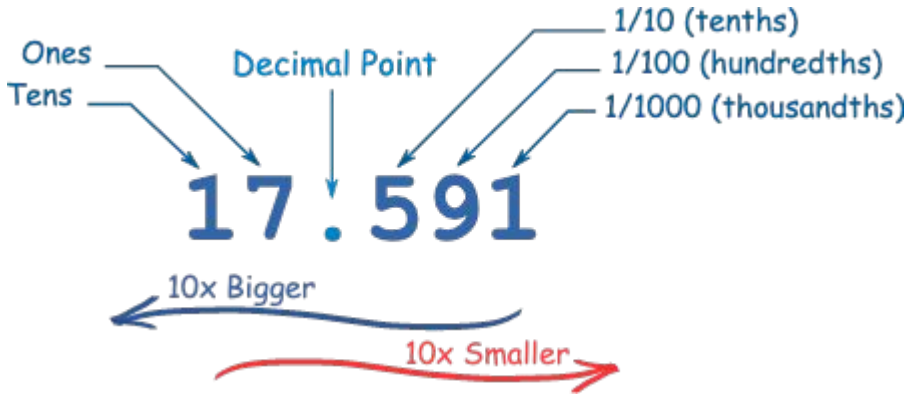
- time_millis
- inadequate dates

```
63
64  SELECT
65        min(                    ) AS min_date1,
66        max(                    ) AS max_date1
67    FROM
68
```

MIN_DATE1	MAX_DATE1
0006-05-12	2580-05-15

Data type issues

- time_millis
- inadequate dates
- decimal



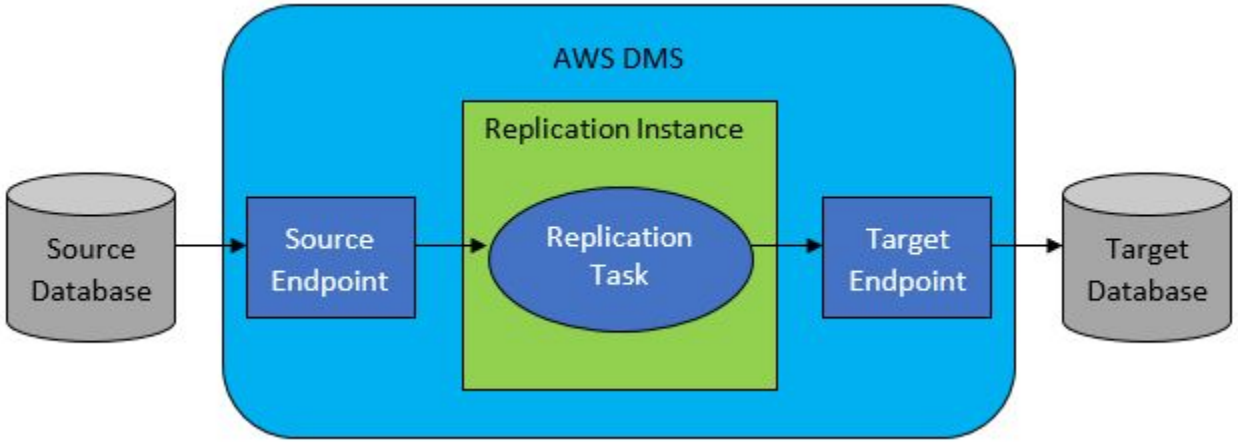
Data type issues: solution

Data type issues: solution

DMS: Source Endpoint

```
DataSourceEndpoint:
  Type: AWS::DMS::Endpoint
  Properties:
    EndpointIdentifier: prod-data-source-endpoint
    EngineName: data_source
    EndpointType: source
    Username: '{{resolve:secretsmanager:service:SecretString:username}}'
    Password: '{{resolve:secretsmanager:service:SecretString:password}}'
    ServerName: '{{resolve:secretsmanager:service:SecretString:host}}'
    Port: '{{resolve:secretsmanager:service:SecretString:port}}'
    DatabaseName: '{{resolve:secretsmanager:service:SecretString:database}}'
    ExtraConnectionAttributes: 'executeTimeout=3600;'
```

DMS: ReplicationTask fun part



DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

DataSource schema: my_schm

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

DataSource schema: my_schm

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

```
In [11]: len('my_schm')
Out[11]: 7
```

DataSource schema: my_schm

known issue when using DB2 as source with schema name having (_) underscore and less than 8 characters selection rule fails

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

```
In [11]: len('my_schm')
Out[11]: 7
```

DataSource schema: my_schm

known issue when using DB2 as source with schema name having (_) underscore and less than 8 characters selection rule fails

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

```
In [11]: len('my_schm')
Out[11]: 7
```

DataSource schema: my_schm

known issue when using DB2 as source with schema name having (_) underscore and less than 8 characters selection rule fails

We are in process of updating the documentation with this limitation.

Being said that there is a workaround to select one specific table by changing "rule-action" from "include" to "explicit"

DMS: ReplicationTask fun part

No tables were found at task initialization. Either the selected table(s) no longer exist or no match was found for the table selection pattern

```
In [11]: len('my_schm')
Out[11]: 7
```

DataSource schema: my_schm

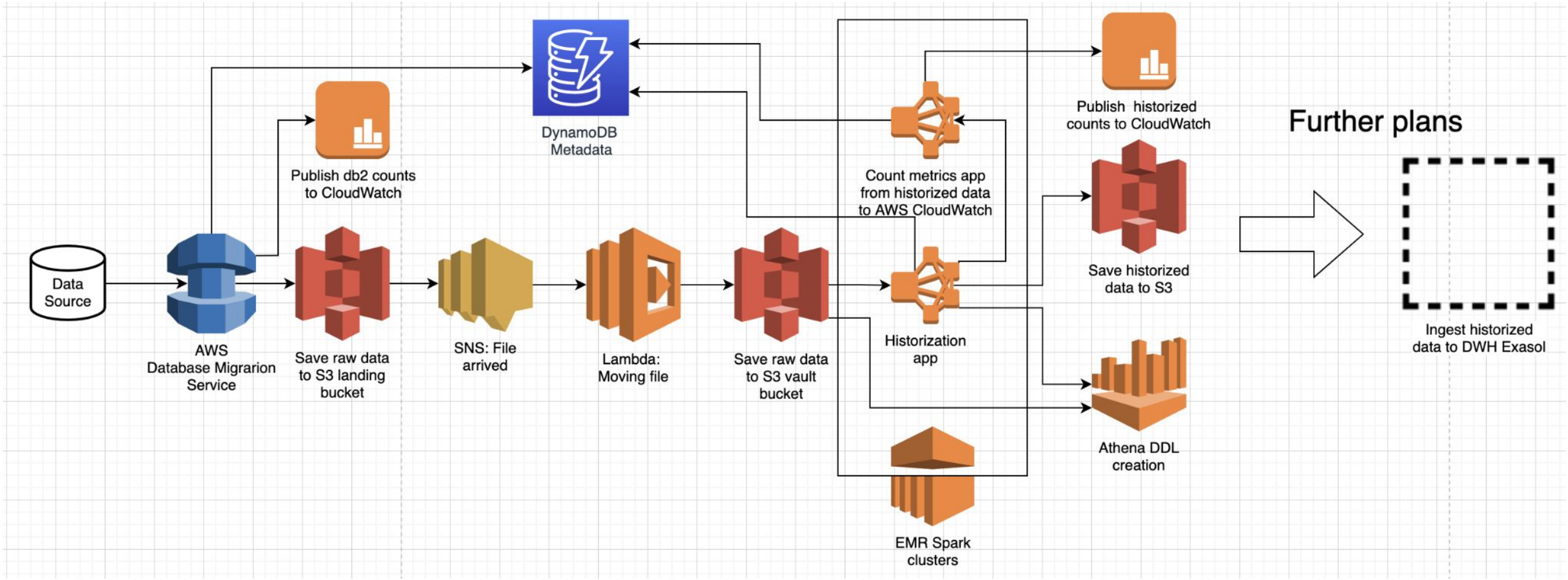
known issue when using DB2 as source with schema name having (_) underscore and less than 8 characters selection rule fails

We are in process of updating the documentation with this limitation.

Being said that there is a workaround to select one specific table by changing "rule-action" from "include" to "explicit"

`rule-action`	`include`, `exclude`, `explicit`	A value that includes or excludes the object or objects selected by the rule. If `explicit` is specified, you can select and include only one object that corresponds to an explicitly specified table and schema.

Further work



Achievements

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake
- We have automated and auto scaled process

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake
- We have automated and auto scaled process
- About 70 tables are running day by day

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake
- We have automated and auto scaled process
- About 70 tables are running day by day
- We have just one entry point for adding new tables to the process - all other steps are catching up and reusing this metadata

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake
- We have automated and auto scaled process
- About 70 tables are running day by day
- We have just one entry point for adding new tables to the process - all other steps are catching up and reusing this metadata
- We enabled process monitoring and alarming on data and process issues

Achievements

- We have built ETL process for full and incremental load data from Data Source to Data Lake
- We have automated and auto scaled process
- About 70 tables are running day by day
- We have just one entry point for adding new tables to the process - all other steps are catching up and reusing this metadata
- We enabled process monitoring and alarming on data and process issues
- Our data stored in effective way and easy to access with different tools

Learning points to you

Q&A: Your voice matters

Link to the anonymous survey

Metadata in DynamoDB

Partition key:
TableID is the following composition **<Source>-<Target>-<SchemaName>-<TableName>**

Metadata in DynamoDB

Partition key:
TableID is the following composition **<Source>-<Target>-<SchemaName>-<TableName>**

Sort key: **TableName**

Metadata in DynamoDB

Partition key:
TableID is the following composition **<Source>-<Target>-<SchemaName>-<TableName>**

Sort key: **TableName**