**What is JSON?**

JSON, or JavaScript Object Notation, is a minimal , readable format for structuring data. It is used primarily to transmit data between a server and web application, as alternative to XML.

**Keys and Values**

**The two primary parts that make up JSON are keys and values. Together they make a Key/Value pair.**

**Key: A key is always a string enclosed in quotation marks.**

**Value: A value can be string , number , Boolean expression, array , or object.**

**Key/Value Pair: A key value pair follows a specific syntax, with the key followed by a colon followed by the value. Key/Value pairs are separated by comma.**

**Types of Values:**

* **Array: An associative array of values.**
* **Object: An associative array of key/value pairs.**
* **String: Serval plain text characters which usually from word**
* **Number: An Integer**
* **Boolean: True or False**

**Objects:**

* **JSON objects are surrounded by curly braces{}**
* **JSON objects are written in key/value pairs**
* **Keys and values are separated by a colon:**
* **Each key/value pair is separated by a comma,**
* **Keys must be strings, values must be valid JSON data type.**

**Arrays:**

* **JSON arrays are surrounded by brackets[]**
* **JSON arrays are accessed through its index.. starting at 0**
* **Valid JSON data types.**

**Traversing JSON in Snowflake**

**“applicationData”:{**

**“application”:{**

**“applicationName”:”Order”,**

**“applicationVersion”:”Order Version1”,**

**},**

**“documentId”: “212121221”,**

**“injectionDate”: 2021-10-31”,**

**“status”: “current”,**

**}**

**To traverse a JSON query: <column>:<level1\_element>.<level2\_element>.<level3\_element>**

**Snowflake query example:**

**Raw\_json:applicationData.application.applicationName=Order**

**Raw\_josn:applicationData.application.applicationVersion=Order Version1**

**Load Continuous Data into Snowflake using Snow pipe**

**What problem are we solving ?**

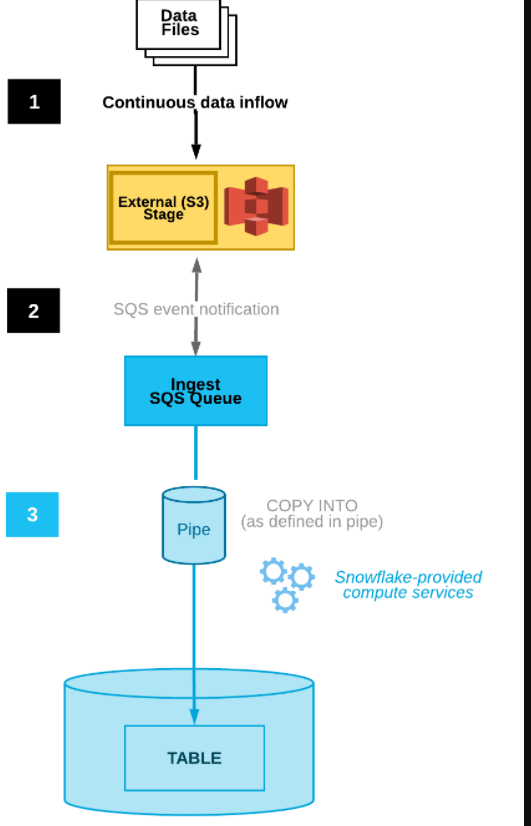
* **You need to ingest data into snowflake and make it available to users at low latency.**
* **You need to automate the ingestion process with minimal administration overhead.**

**We can solve above those problems using Continuous data Mechanism in Snowflake by using Snow Pipe.**

**Introduction:**

**Snowpipe is a mechanism provided by Snowflake to load high frequency or streaming data. Snowpipe provides us with capability to load data as soon as it becomes available in a defined stage.**

**Design:**

****

1. **JSON Data is Landed into an external stage (AWS S3)**
2. **A Simple Queue Service (SQS) is triggered from S3.**
3. **A pipe reads from the queue and loads data into a the target table**

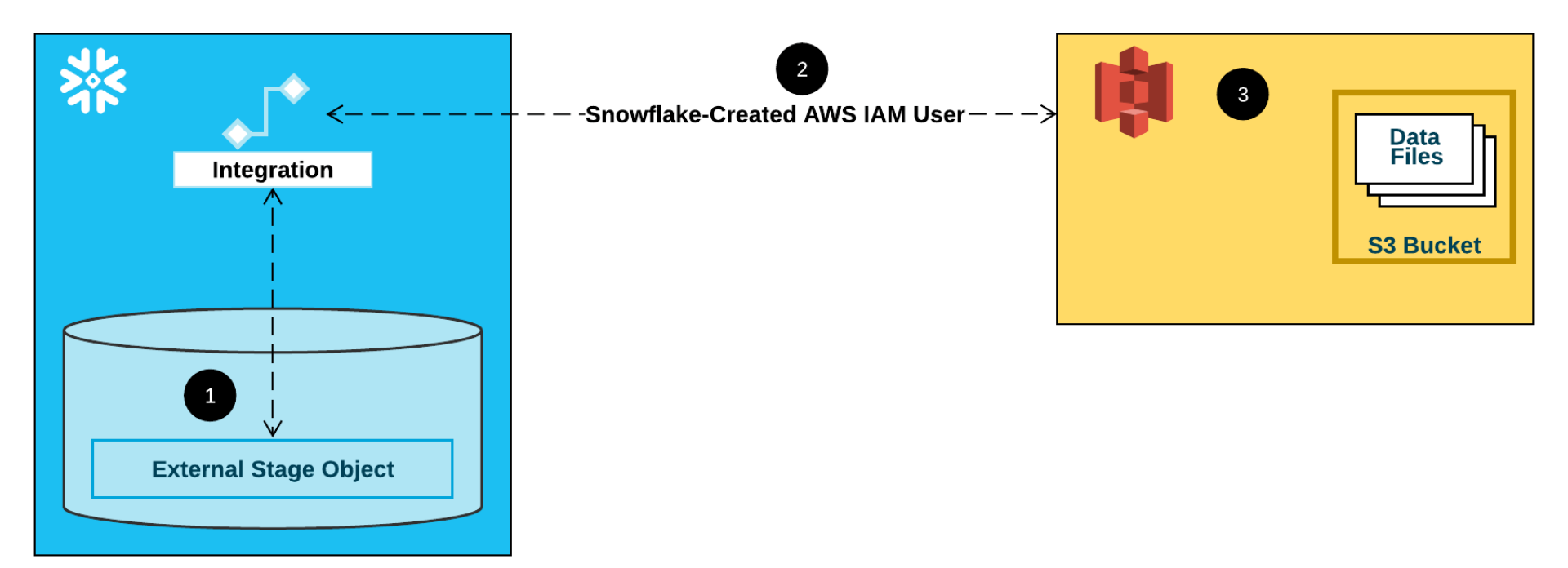
**Steps Involved:**

* **Create a external stage (using storage integration)**
* **Create a pipe with auto-ingest enabled**
* **Configure event notification**
* **Monitor snowflake Pipe run status and See table if the data is loaded in a Snowflake table or not.**

**Creating External Stage:**

**Creating a storage integration object means we can attach this to AWS IAM user and encapsulates the credentials needed to access our AWS resources.**

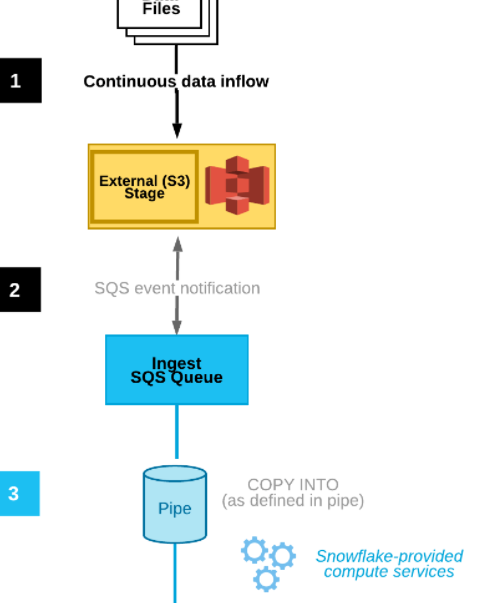
**An external stage can then be created from storage integration object.**

****

**Creating a Pipe:**

**Next, we create a pipe - which is a COPY INTO statement wrapped in a CREATE PIPE command.**

**This creates a SQS queue in the background on AWS for us. We use show pipes to get the Amazon Resource Number(ARN) for this and use it when configuring events for our S3 bucket.**

****

**Creating Target Table:**

**Note: Here, the pipe definition is created, this will not result automatically loading of any data into the target table until event has been defined in the cloud storage.**

**Finally , you can load new file in your specified path in the s3 bucket . Run the following SQL to check if the uploaded file is in your stage location.**

**LS @<stage\_name>;**

**Wait for 10-15 seconds and verify that the data was loaded into the target table (i.e the table identified in the COPY statement in your pipe definition)**

**You can validate by checking the row count of the table.**

**Pipe status:**

**Check the pipe status using SQL `select SYSTEM$PIPE\_STATUS(’<pipe\_name>’);**

**File error:**

**Check whether the pipe and copy statement within it produced an error while processing the file. Only first error within the file is logged. It will show the exact filename and path that caused the error.**

**SQL:**

**select \***

**from table(information\_schema.copy\_history(table\_name=>'REGION', start\_time=> dateadd(day, -7,**

**current\_timestamp())));**

**select \* from table(validate\_pipe\_load(**

**pipe\_name=>'MYPIPE',**

**start\_time=>dateadd(hour, -1, current\_timestamp())));**

**You can check Snow pipe ingestion audit logs in COPY\_HISTORY also. This will also reveal any errors that the COPY statement encountered. COPY\_HISTORY is limiting as it spans only the past 14 days and .**