Week I

Chapter I

Data Engineer

Trainer: Balazs Balogh

CUBIX

INSTITUTE OF TECHNOLOGY

Data Modeling - Data warehouse vs data lake

- A data warehouse is a structured, optimized database designed for analytical queries.
- A data lake is a vast storage repository that holds raw, unstructured data.
- Data warehouses focus on processing and analyzing historical data, providing a structured view for reporting, while data lakes store diverse data types in their raw form, accommodating a wider range of data sources.
- In AWS Data Lake is S3, Data Warehouse is Amazon Redshift.

Data Modeling - OLTP vs. OLAP databases

- OLTP (Online Transaction Processing) involves managing day-to-day transactional operations in real-time, focusing on quick, individual record updates.
- OLAP (Online Analytical Processing) deals with complex queries and data analysis, providing a multidimensional view of aggregated data.
- OLTP systems are transactional databases, while OLAP systems support analytical queries for business intelligence.
- OLAP: Amazon Redshift (specifically designed for OLAP), Amazon RDS (Relational Database Service), Amazon Aurora
- OLTP: Amazon RDS, Amazon Aurora

Data Modeling – Dimension / Fact / Mapping tables

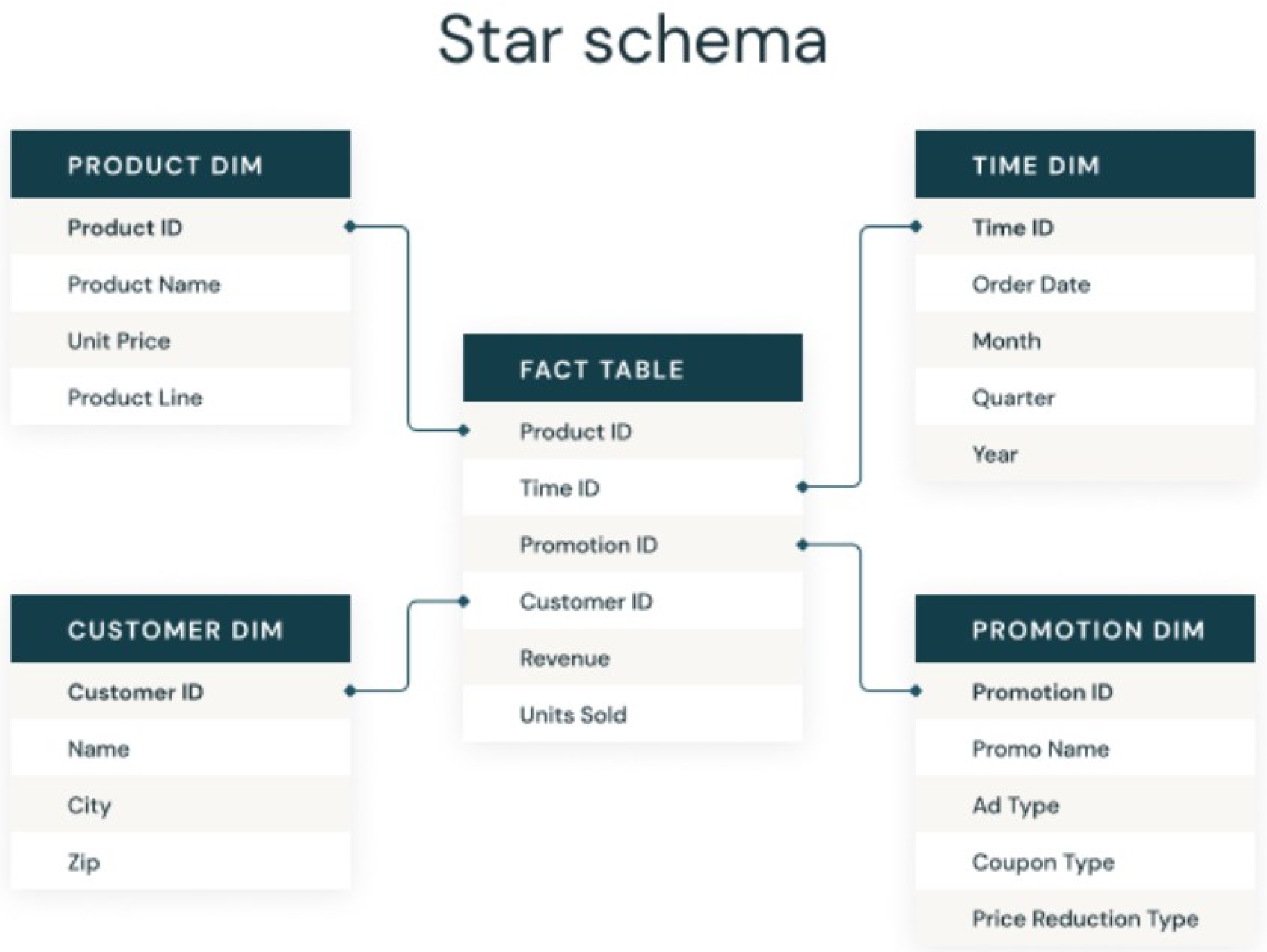
- In a data warehouse, dimension tables store descriptive attributes.
- Fact tables contain numeric performance metrics
- Mapping tables establish relationships between dimensions and facts.
- Dimensions provide context to the data, facts quantify business processes, and mapping tables bridge relationships between the two, ensuring data integrity and facilitating analysis.

Data Modeling – Dimension / Fact / Mapping tables

- Dimension tables:
  - Product: Product ID, Product Name, Unit Price, Product Line
  - Customer: Customer ID, Name, City, Zip
  - Time: Time ID, Order Date, Month, Quarter, Year
  - Promotion: Promotion ID, Promo Name, Ad Type, Coupon Type, Price Reduction Type
- Fact table:
  - Product ID, Time ID, Promotion ID, Customer ID, Revenue, Units Sold
- The fact table connects to multiple other dimension tables along "dimensions" like time, or product.

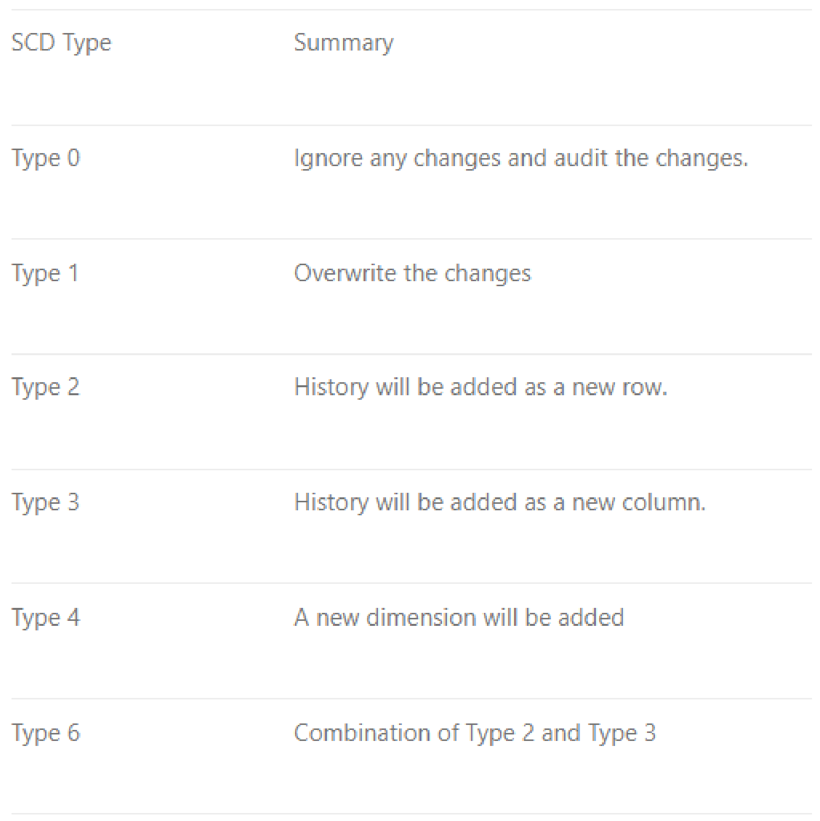
Data Modeling – Star Schema

- A star schema has a single fact table in the center, containing business "facts" (like transaction amounts and quantities). The fact table connects to multiple other dimension tables along "dimensions" like time, or product.



SCD – Slowly Changing Dimension

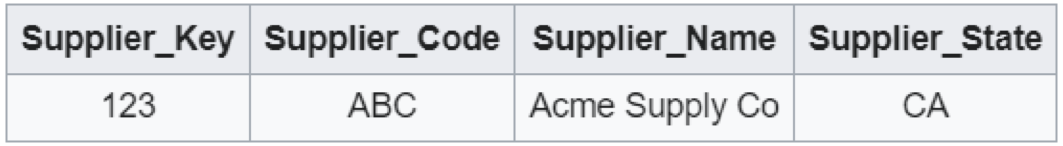
- Slowly Changing Dimensions in Data Warehouse is an important concept that is used to enable the historic aspect of data in an analytical system.
- SCD is used to track changes in the data. There are six common types of it.



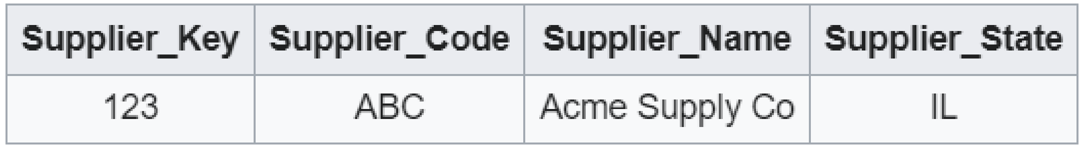
SCD Type	Summary
Type 0	Ignore any changes and audit the changes.
Type 1	Overwrite the changes
Type 2	History will be added as a new row.
Type 3	History will be added as a new column.
Type 4	A new dimension will be added
Type 6	Combination of Type 2 and Type 3

SCD – Slowly Changing Dimension

- Type 0: The attributes never change, like date of birth. Most data tables are type 0, no change tracking needed.
- Type 1: This method will overwrite old data, therefore this isn’t tracking changes aswell.
  - In the above example, Supplier_Code is the natural key and Supplier_Key is a surrogate key. Technically, the surrogate key is not necessary, since the row will be unique by the natural key.



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA



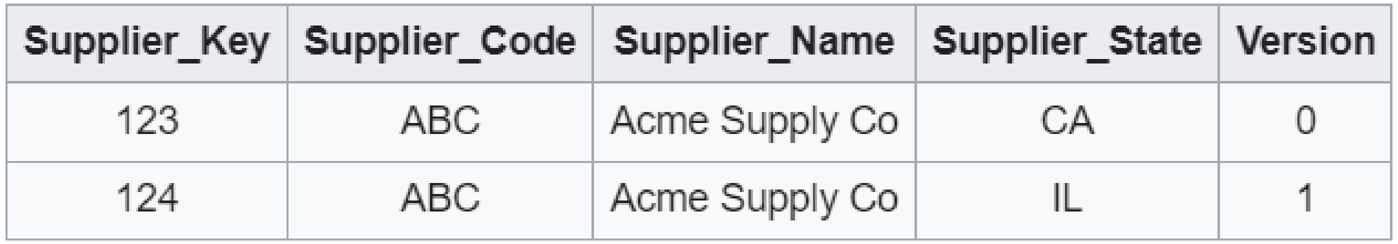
Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	IL

SCD – Slowly Changing Dimension

- Type 2: One of the most popular methods. From Type 2, it is tracking the changes.
- We can use multiple ways of it:
  - versioning
  - start / end date
  - effective date / current flag

SCD – Slowly Changing Dimension

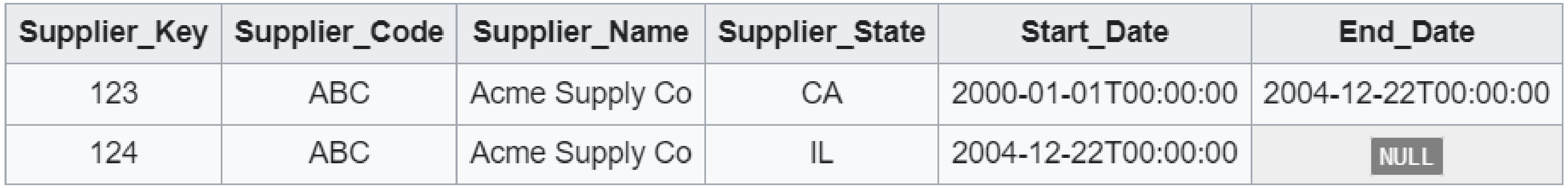
- versioning:
  - Notice that we are using the Supplier_Key and Supplier_Code for tracking.
  - When the data changes, the old record is updated with Version = 0, and a new row is inserted with a new Supplier_Key with the change (CA to IL) and the Version = 1
  - When you want to get only the active records you can use
    - SELECT * FROM table WHERE Version = 1



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Version
123	ABC	Acme Supply Co	CA	0
124	ABC	Acme Supply Co	IL	1

SCD – Slowly Changing Dimension

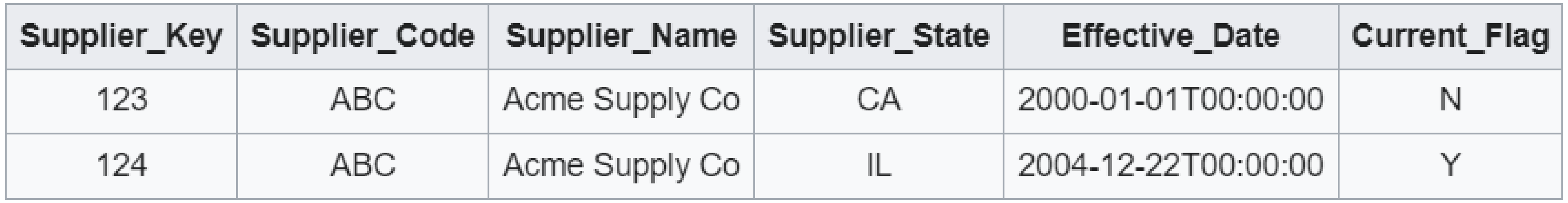
- start / end date:
  - When the Supplier_State changes, End_Date will be the time and date of the change, and in the next row with a new Supplier_Key it will be the Start_Date.
  - The current row has an End_Date of NULL, or 9999-12-31.
  - With the SELECT * FROM table WHERE End_Date IS NULL it's easy to look for the active rows.



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date
123	ABC	Acme Supply Co	CA	2000-01-01T00:00:00	2004-12-22T00:00:00
124	ABC	Acme Supply Co	IL	2004-12-22T00:00:00	NULL

SCD – Slowly Changing Dimension

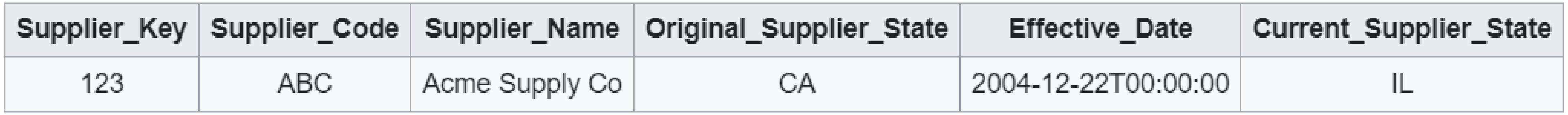
- effective date / current flag:
  - The Current_Flag value of “Y” indicates the current version.
  - When a change is made, the old record gets updated with Current_Flag = ‘N’ and a new row will be inserted with a new Supplier_Key and the change (CA to IL) with the current time and date, and a “Y” as the Current_Flag.



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Effective_Date	Current_Flag
123	ABC	Acme Supply Co	CA	2000-01-01T00:00:00	N
124	ABC	Acme Supply Co	IL	2004-12-22T00:00:00	Y

SCD – Slowly Changing Dimension

- Type 3: This method adds a new attribute.
- It preserves limited history as it is limited to the number of columns designated for storing historical data.

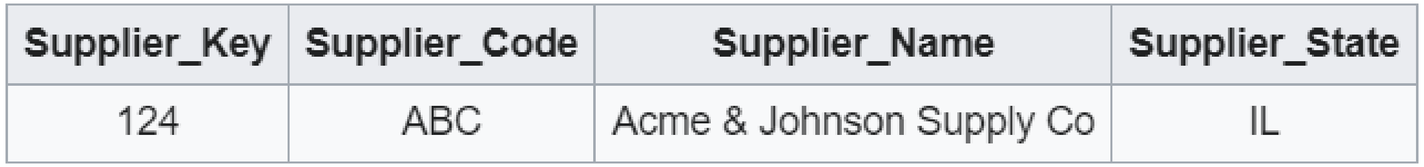


Supplier_Key	Supplier_Code	Supplier_Name	Original_Supplier_State	Effective_Date	Current_Supplier_State
123	ABC	Acme Supply Co	CA	2004-12-22T00:00:00	IL

SCD – Slowly Changing Dimension

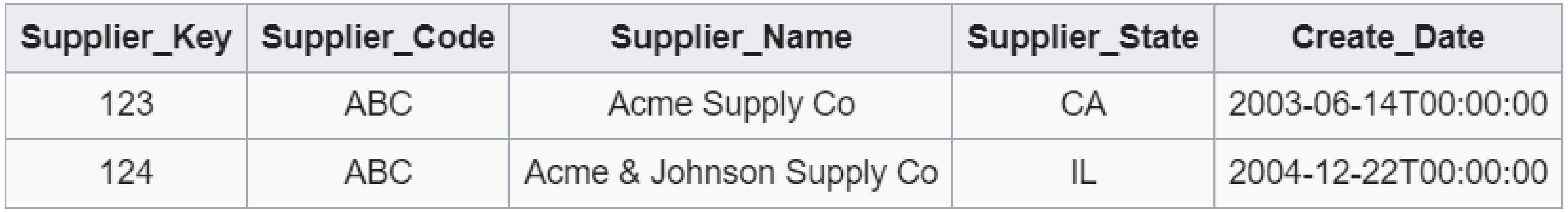
- Type 4: This method uses history tables to preserve changes.
- One table holds only the actual values, and the history table keeps the changes.

Supplier



Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
124	ABC	Acme & Johnson Supply Co	IL

Supplier_History



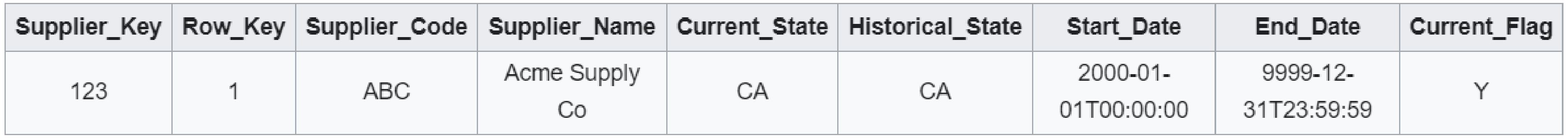
Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Create_Date
123	ABC	Acme Supply Co	CA	2003-06-14T00:00:00
124	ABC	Acme & Johnson Supply Co	IL	2004-12-22T00:00:00

SCD – Slowly Changing Dimension

- Type 5: Type 5 is an enhancement to Type 4 that incorporates elements of Type 1.
- It allows the current view of a dimension to be embedded into another dimension.
- Type 5 created a snowflake schema and kept the current view as an addon dimension.

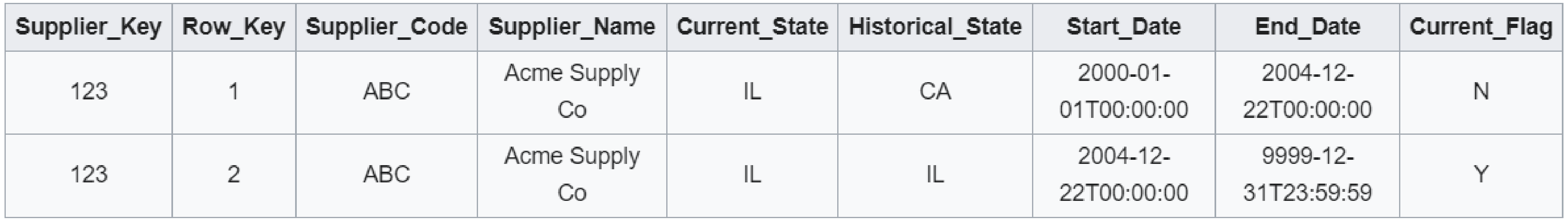
SCD – Slowly Changing Dimension

- Type 6: Combines the approaches of types 1, 2 and 3 (1 + 2 + 3 = 6).
- [illegible]



Supplier_Key	Row_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	1	ABC	Acme Supply Co	CA	CA	2000-01-01T00:00:00	9999-12-31T23:59:59	Y

- When the Current_State changes from CA to IL, a new record is added with a Row_Key to ensure that we have a unique identifier



Supplier_Key	Row_Key	Supplier_Code	Supplier_Name	Current_State	Historical_State	Start_Date	End_Date	Current_Flag
123	1	ABC	Acme Supply Co	IL	CA	2000-01-01T00:00:00	2004-12-22T00:00:00	N
123	2	ABC	Acme Supply Co	IL	IL	2004-12-22T00:00:00	9999-12-31T23:59:59	Y

AWS S3 – Simple Storage Service

- AWS S3 is a cloud-based object storage service that allows you to store and retrieve any amount of data at any time.
- It provides a scalable and durable solution for storing files, images, and data objects, accessible via a unique URL.
- S3 is commonly used as a foundation for building a data lake.
- Used also for backup, data archiving, hosting static websites, and as a storage backend for various applications.
- The main part of it are the buckets. A bucket is a container for objects. It is similar to a folder.
- An object is an entity stored in a bucket. It can be anything from a text file to an image, video, or other binary data.

AWS S3 – Simple Storage Service

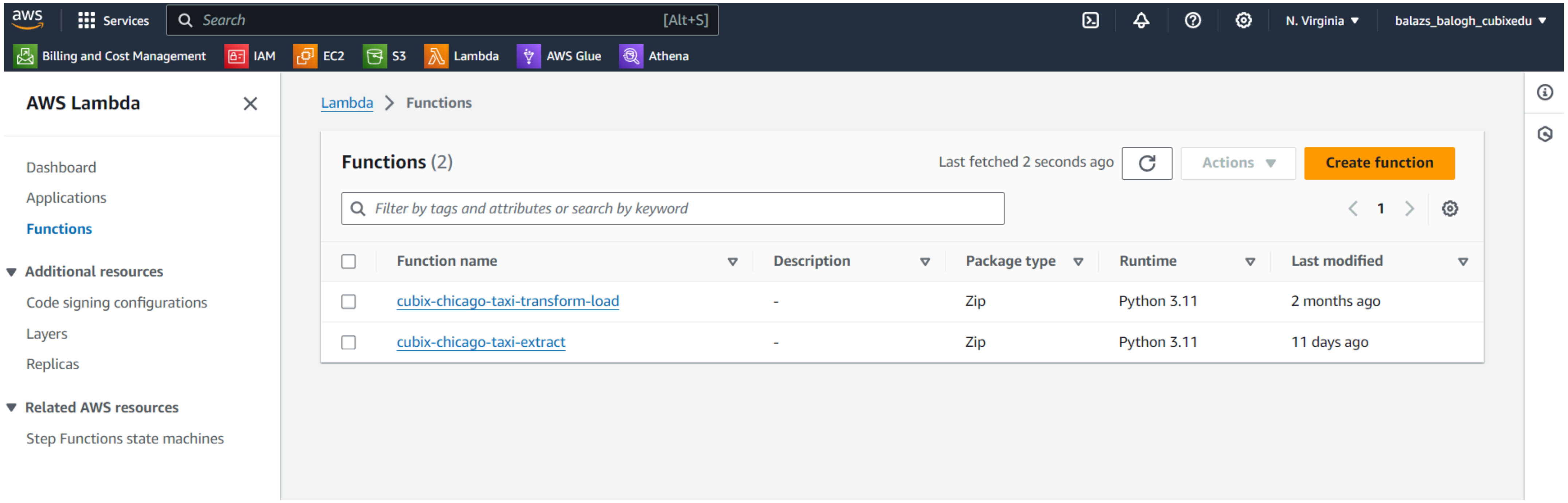
- A key is a unique identifier for an object within a bucket. It is similar to a file path and is used to organize and retrieve objects.
- In the path "photos/2023/january/cat.jpg“, "photos/2023/january/cat.jpg" is the key.
- S3 offers different storage classes to optimize costs and performance based on the access patterns of your data. It ranges from Standard to Glacier.
- In Standard it needs milliseconds to retrieve your data, in Glacier it needs hours, but it is the cheapest solution to store rarely accessed data.

AWS Lambda

- AWS Lambda is a serverless compute service that lets you run your code without provisioning or managing servers.
- With Lambda, you can execute functions in response to events like changes to data in an S3 bucket, incoming HTTP requests, or updates to a database.
- It scales automatically, charges only for the compute time consumed, and is well-suited for building event-driven, microservices, and serverless architectures.

AWS Lambda – Extract function

- Home screen:

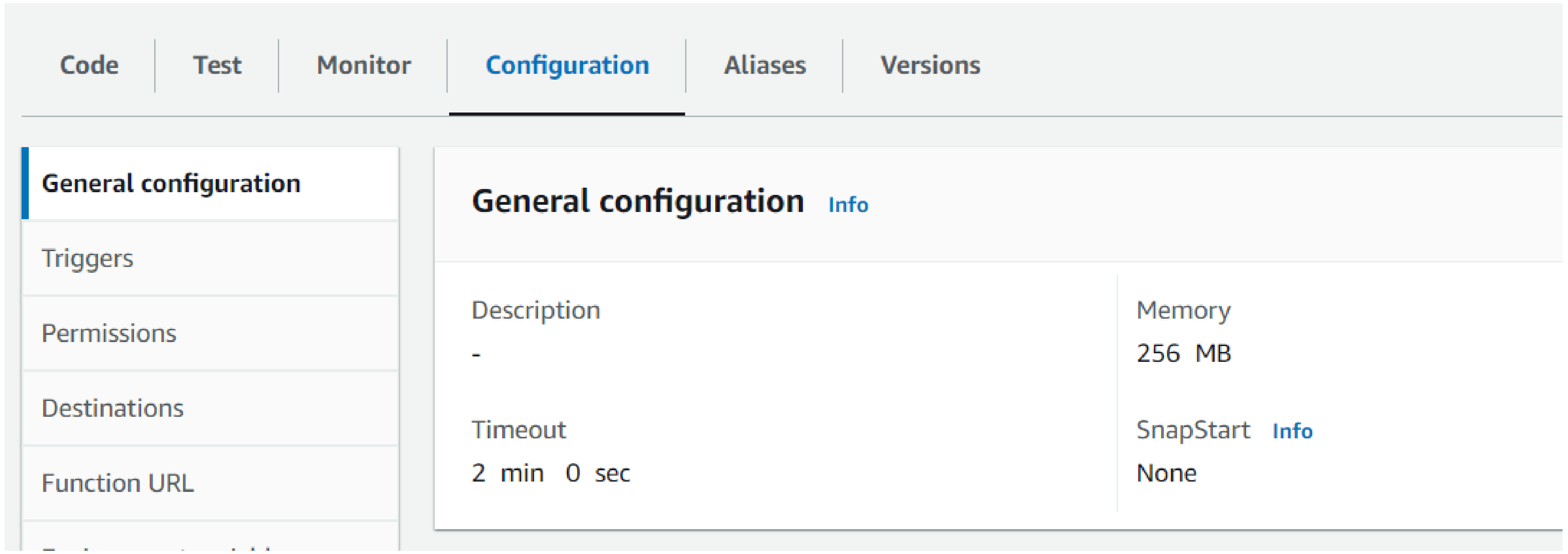


Functions (2)

	Function name	Description	Package type	Runtime	Last modified
☐	cubix-chicago-taxi-transform-load	-	Zip	Python 3.11	2 months ago
☐	cubix-chicago-taxi-extract	-	Zip	Python 3.11	11 days ago

AWS Lambda – Extract function

- Change the Timeout to 2 minutes, and the memory to 256 MB.



General configuration

Description
-

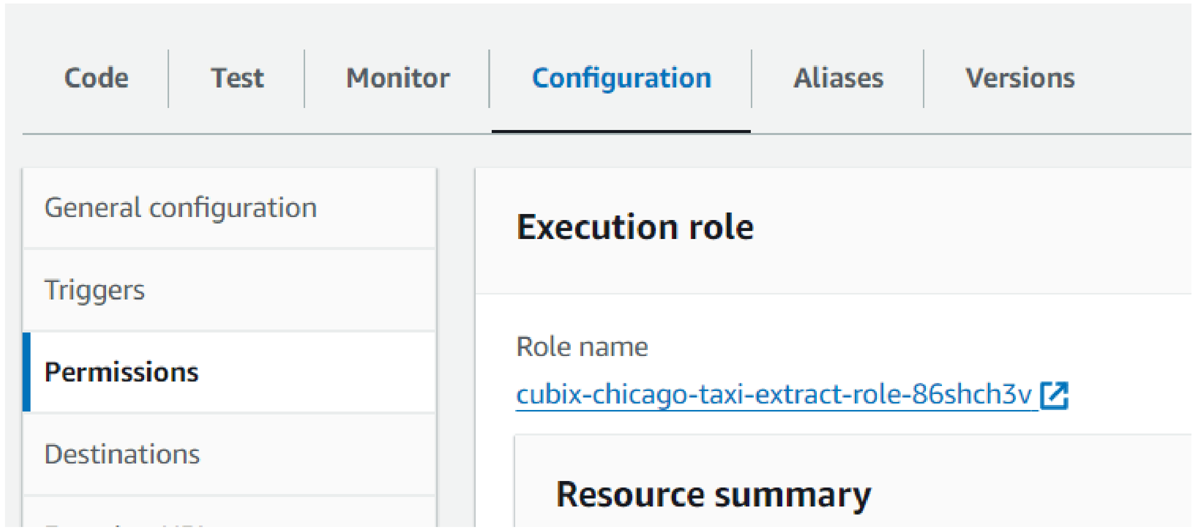
Timeout
2 min 0 sec

Memory
256 MB

SnapStart
None

AWS Lambda – Extract function

- Change the Execution Role's permission

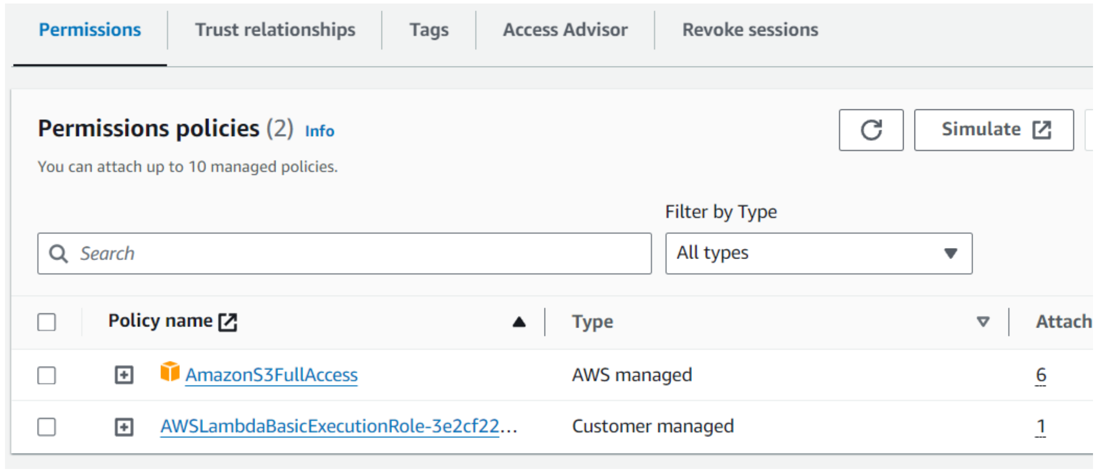


Execution role

Role name

cubix-chicago-taxi-extract-role-86shch3v

Resource summary



Permissions policies (2) Info

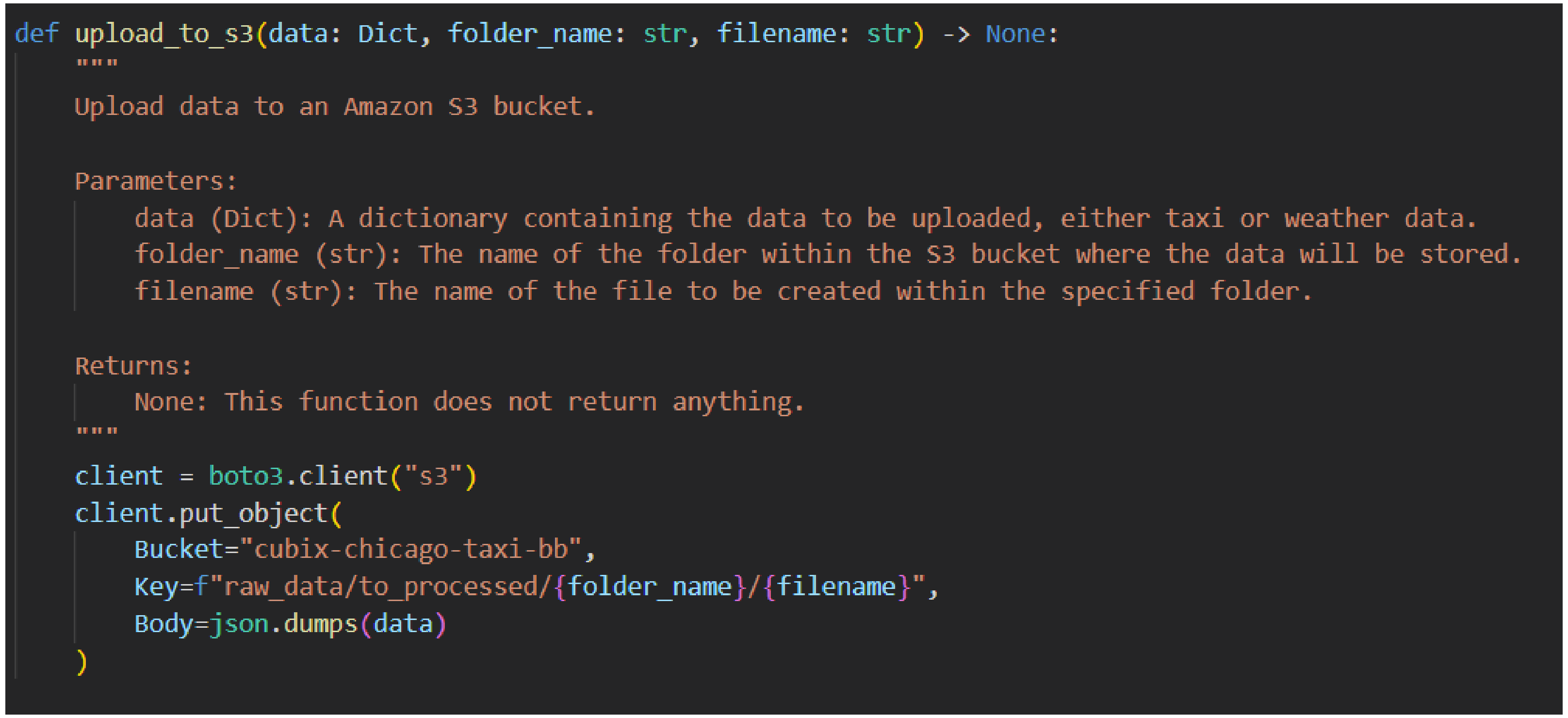
You can attach up to 10 managed policies.

Policy name	Type	Attach
AmazonS3FullAccess	AWS managed	6
AWSLambdaBasicExecutionRole-3e2cf22...	Customer managed	1

- Add AmazonS3FullAccess to be able to save data to your bucket.

Docstrings

- Python docstrings are the string literals that appear right after the definition of a function, method, class, or module. They are used to document our code.
- Give a short description about the function, the parameters, and the return value.



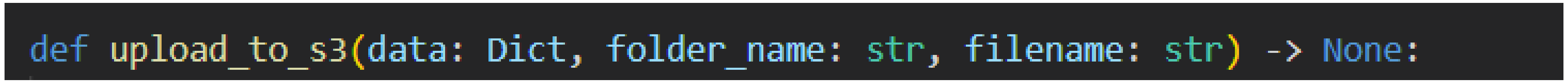
```python
def upload_to_s3(data: Dict, folder_name: str, filename: str) -> None:
    """
    Upload data to an Amazon S3 bucket.

    Parameters:
        data (Dict): A dictionary containing the data to be uploaded, either taxi or weather data.
        folder_name (str): The name of the folder within the S3 bucket where the data will be stored.
        filename (str): The name of the file to be created within the specified folder.

    Returns:
        None: This function does not return anything.
    """
    client = boto3.client("s3")
    client.put_object(
        Bucket="cubix-chicago-taxi-bb",
        Key=f"raw_data/to_processed/{folder_name}/{filename}",
        Body=json.dumps(data)
    )
```

Type hints

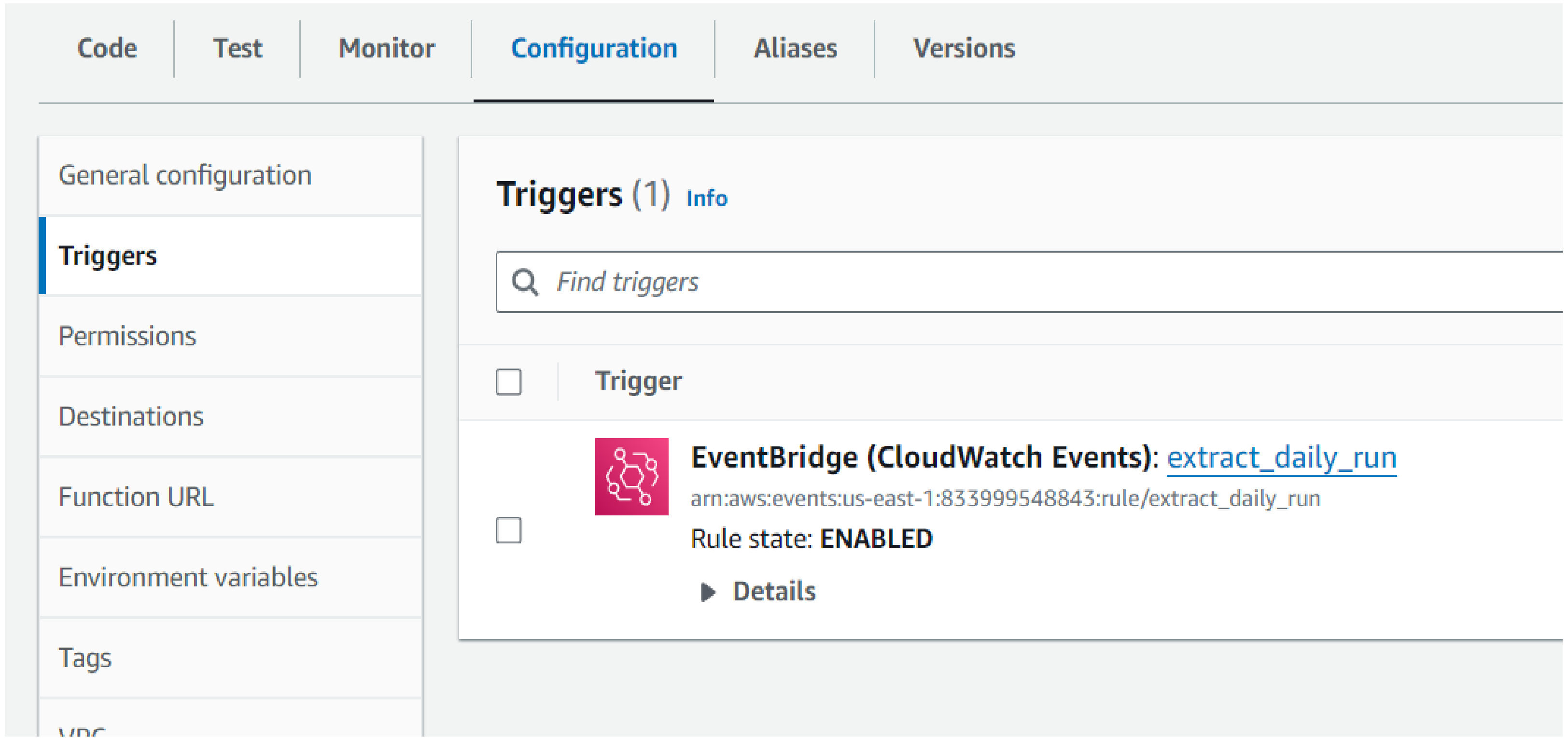
- Type hinting is a formal solution to statically indicate the type of a value within your Python code.
- It makes easier to other developers (or the future you) to understand your code.
- In the below function, we have a data parameter which is a dictionary, and folder_name and file_name as string. It does not return anything, so the value is None.



```python
def upload_to_s3(data: Dict, folder_name: str, filename: str) -> None:
```

AWS Lambda - Automation

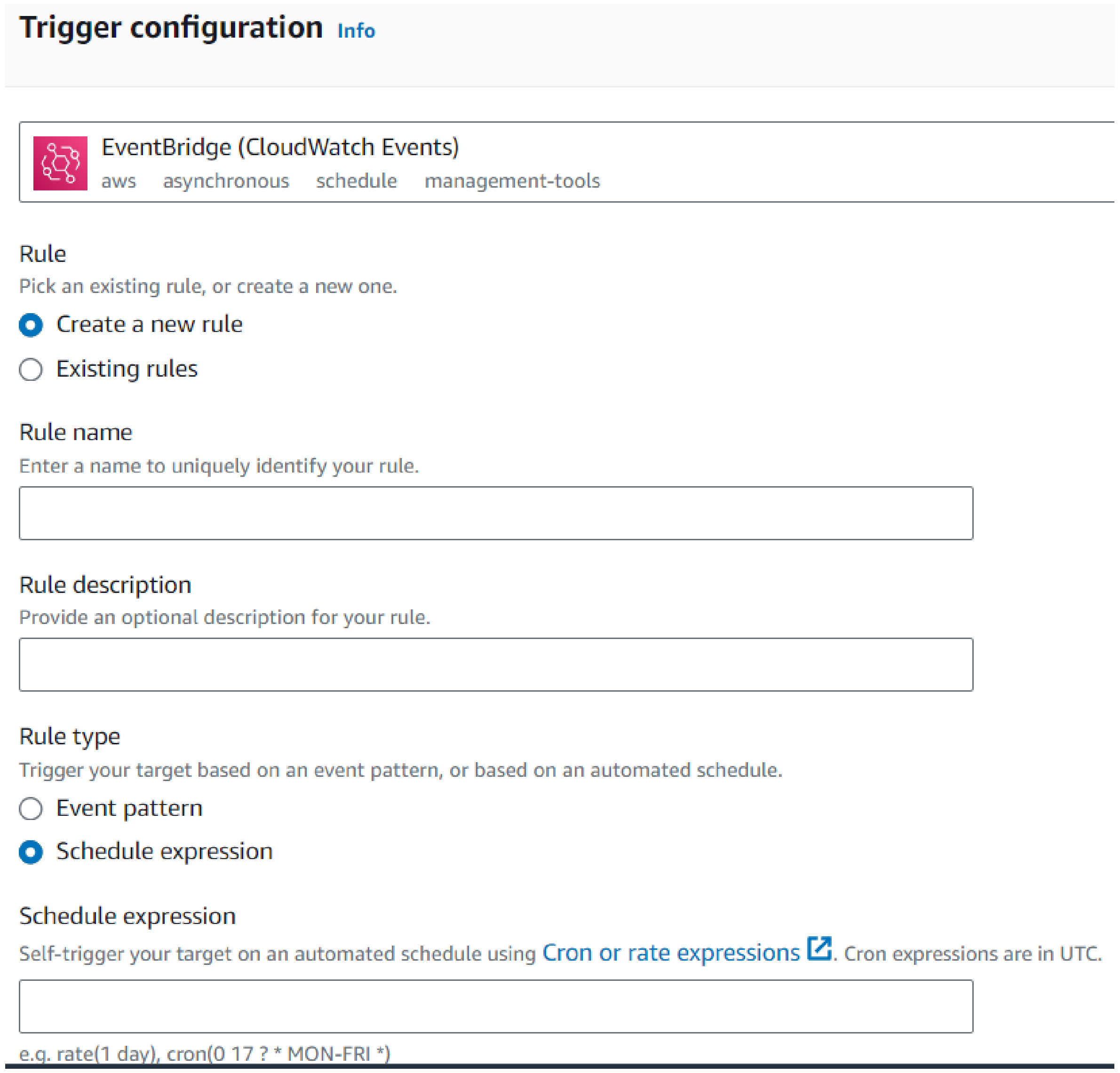
- Choose Triggers under Configuration to set up a schedule for your function.
- Click on Add trigger



EventBridge (CloudWatch Events): extract_daily_run
arn:aws:events:us-east-1:833999548843:rule/extract_daily_run
Rule state: ENABLED
Details

AWS Lambda - Automation

- Choose EventBridge (CloudWatch Events).
- Fill in the name and the description.
- For the Schedule expression, type “1 day”, which means that it will run daily at the same time when you first applied it.



Trigger configuration Info

EventBridge (CloudWatch Events)
aws asynchronous schedule management-tools

Rule
Pick an existing rule, or create a new one.
[✓ Create a new rule] Existing rules

Rule name
Enter a name to uniquely identify your rule.

Rule description
Provide an optional description for your rule.

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
Event pattern [✓ Schedule expression]

Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

CUBIX INSTITUTE OF TECHNOLOGY