In [23]:
```
from pyspark.sql import SparkSession

spark = (
    SparkSession
    .builder
    .master("local[*]")
    .appName("Spark Streaming")
    .getOrCreate()
)

spark
```

Out[23]:

SparkSession - in-memory

SparkContext

Spark UI

Version	v3.3.0
Master	local[*]
AppName	Spark Streaming

In [24]:
```
spark.conf.set('spark.sql.streaming.schemaInference', True)

streaming_df = (
    spark
    .readStream
    .option('cleanSource', 'off') # off(default) / archive / delete
    .option('maxFilesPerTrigger', 1)
    .json("datas/", multiLine=True)
)
```

In [25]:
```
streaming_df.printSchema()
```

```
root
 |-- _corrupt_record: string (nullable = true)
 |-- customerId: string (nullable = true)
 |-- data: struct (nullable = true)
 |    |-- devices: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- deviceId: string (nullable = true)
 |    |    |    |-- measure: string (nullable = true)
 |    |    |    |-- status: string (nullable = true)
 |    |    |    |-- temperature: long (nullable = true)
 |-- eventId: string (nullable = true)
 |-- eventOffset: long (nullable = true)
 |-- eventPublisher: string (nullable = true)
 |-- eventTime: string (nullable = true)
```

In [26]:
```
from pyspark.sql.functions import explode

explode_df = streaming_df.withColumn("devices", explode("data.devices"))
```

In [27]:
```
explode_df.printSchema()
```

```
root
 |-- _corrupt_record: string (nullable = true)
 |-- customerId: string (nullable = true)
 |-- data: struct (nullable = true)
 |    |-- devices: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- deviceId: string (nullable = true)
 |    |    |    |-- measure: string (nullable = true)
 |    |    |    |-- status: string (nullable = true)
 |    |    |    |-- temperature: long (nullable = true)
 |-- eventId: string (nullable = true)
 |-- eventOffset: long (nullable = true)
 |-- eventPublisher: string (nullable = true)
 |-- eventTime: string (nullable = true)
 |-- devices: struct (nullable = true)
 |    |-- deviceId: string (nullable = true)
 |    |-- measure: string (nullable = true)
 |    |-- status: string (nullable = true)
 |    |-- temperature: long (nullable = true)
```

In [28]:
```
from pyspark.sql.functions import col

flatened_df = (
    explode_df
    .drop('data')
    .withColumn('deviceId', col('devices.deviceId'))
    .withColumn('measure', col('devices.measure'))
    .withColumn('status', col('devices.status'))
    .withColumn('temperature', col('devices.temperature'))
)

flatened_df.printSchema()
```

```
root
 |-- _corrupt_record: string (nullable = true)
 |-- customerId: string (nullable = true)
 |-- eventId: string (nullable = true)
 |-- eventOffset: long (nullable = true)
 |-- eventPublisher: string (nullable = true)
 |-- eventTime: string (nullable = true)
 |-- devices: struct (nullable = true)
 |    |-- deviceId: string (nullable = true)
 |    |-- measure: string (nullable = true)
 |    |-- status: string (nullable = true)
 |    |-- temperature: long (nullable = true)
 |-- deviceId: string (nullable = true)
 |-- measure: string (nullable = true)
 |-- status: string (nullable = true)
 |-- temperature: long (nullable = true)
```

In [30]:
```
flatened_df = flatened_df.drop('devices')
```

In [ ]:

```
(
    flatened_df
    .writeStream
    .format("csv")
    .option("path", "outs/json_result_out.csv")
    .option('checkpointLocation', 'checkpoint_dir')
    .start()
    .awaitTermination()
)
```