Event-based Scheduler

How the scheduler works

The scheduler uses scheduling configurations to know if a job must be triggered.

A scheduling configuration is a **set of dependencies**. **All dependencies** for a scheduling configuration must be met, to **trigger a job**.
Triggering a job does not invalidate dependencies of the scheduling configuration. *(for more details about that take a look at the exercise scheduling configurations)*

Everytime you receive **an event that matches one of your configuration dependencies**, you will check **if all dependencies are met**. If that's the case, you will **trigger a job**.

A scheduling configuration is as follows :

```
{
 "name": "string",
 "dependencies": [
   {
     "type": "string",
     "resourceId": "string",
     "lifeDuration": "integer"
   }
 ]
}
```

- `name`: name of the configuration
- `dependencies`: is an array of dependency :
    - `type`: can be one of `[ FILE, TIME_BASED, TABLE ]`
    - `resourceId`: is a string that represents the resourceId of the event validating the dependency
    - `lifeDuration`: number of seconds that the dependency is valid when validated by an event.

Events will be sent to the scheduler with the following **specifications** :

```
{
 "eventType": "string",
 "eventTimestamp": "string",
 "eventResourceId": "string"
```

```
}
```

- eventType : can be one of `[ FILE, TIME_BASED, TABLE ]`
- eventTimestamp : is an ISO 8601 timestamp in the format %Y-%m-%dT%H:%M:%S.%fZ (https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes) that represents the timestamp
- eventResourceId: is a string that represents the resourceId of the event

The scheduler is stateful. For each scheduling configuration, it must store the last job execution.

To **validate** a **dependency**:

- event "`eventType`" must match dependency "`type`"
- event "`eventResourceId`" must match dependency "`resourceId`"
- event "`eventTimestamp`" must be greater than the last job execution timestamp for the given scheduling configuration*

Since the **scheduler is **event based**, be aware that **events** can be **received** in **any order.***

Each **event** that **satisfies** a scheduling configuration **dependency must be stored**.

Exercice : *Implement the scheduler defined above*

You will write a Flask application that exposes a single HTTP endpoint accepting POST requests.

This endpoint will only accept `application/json` MIME media type.

The accepted payload will correspond to the events' JSON specifications described above.

You can use whichever storage layer you like, but make sure that it is easily reproducible. (for instance, provide the Docker file if you use a docker image).

To trigger a job, you can simply log a message in this exercise.

Here are the scheduling configurations that you will use :

SCHEDULING_CONFIGURATION_1

*This first scheduling configuration will be triggered **every hour** (dependency TIME_BASED + life duration on FILE dependency), **if** at least one event validating the dependency relative to the FILE has been received during the last 3600s preceding the reception of the TIME_BASED event.*

Notice the lifeDuration of 0 for the TIME_BASED dependency, meaning that if any event validating the FILE dependency has been received in the last 3600 seconds, receiving a TIME_BASED event will instantly trigger the job.

```
{
 "name": "SCHEDULING_CONFIGURATION_1",
 "dependencies": [
   {
     "type": "FILE",
     "resourceId": "/scheduling_configuraiton_1/directory/path/",
     "lifeDuration": "3600"
   },
   {
     "type": "TIME_BASED",
     "resourceId": "cron",
     "lifeDuration": "0"
   }
 ]
}
```

SCHEDULING_CONFIGURATION_2

This second scheduling configuration will be triggered every day (dependency TIME_BASED + life duration on both TABLE dependencies), if you received events that validated the dependency relative to both BIGQUERY_TABLE_NAME_1 and BIGQUERY_TABLE_NAME_2 during the last 86400s preceding the reception of the TIME_BASED event.
Notice the lifeDuration of 0 for the TIME_BASED dependency, meaning that if both event validating dependencies relative to BIGQUERY_TABLE_NAME_1 and BIGQUERY_TABLE_NAME_2 have been received for the last 86400 seconds, receiving a TIME_BASED event will instantly trigger the job.

If you receive :

- *an event related to BIGQUERY_TABLE_NAME_1 at Day 1, 1:00 PM*
- *an event related to BIGQUERY_TABLE_NAME_2 at Day 1, 3:00 PM*

Every time you receive an event related to the cron TIME_BASED resource, a job will be triggered until Day 2 at 1:00 PM.

```
{
 "name": "SCHEDULING_CONFIGURATION_2",
 "dependencies": [
   {
     "type": "TABLE",
     "resourceId": "BIGQUERY_TABLE_NAME_1",
     "lifeDuration": "86400"
```

```
    },
    {
      "type": "TABLE",
      "resourceId": "BIGQUERY_TABLE_NAME_2",
      "lifeDuration": "86400"
    },
    {
      "type": "TIME_BASED",
      "resourceId": "cron",
      "lifeDuration": "0"
    }
  ]
}
```

SCHEDULING_CONFIGURATION_3

This third scheduling configuration will be triggered every time an event relative to BIGQUERY_TABLE_NAME_4 is received and an event relative to BIGQUERY_TABLE_NAME_3 has been received in the last 86400s.

Note that if you receive an event relative to BIGQUERY_TABLE_NAME_3 at Day 1, 1:00PM, every time you will receive an event relative to BIGQUERY_TABLE_NAME_4 until Day 2, 1:00PM the scheduler configuration will be validated, and a job will be triggered.

```
{
 "name": "SCHEDULING_CONFIGURATION_3",
 "dependencies": [
   {
     "type": "TABLE",
     "resourceId": "BIGQUERY_TABLE_NAME_3",
     "lifeDuration": "86400"
   },
   {
     "type": "TABLE",
     "resourceId": "BIGQUERY_TABLE_NAME_4",
     "lifeDuration": "0"
   }
 ]
}
```

Scenario :

Here is an example scenario to help you understand.

About **SCHEDULING_CONFIGURATION_1**

```
{
 "name": "SCHEDULING_CONFIGURATION_1",
 "dependencies": [
   {
     "type": "FILE",
     "resourceId": "/scheduling_configuraiton_1/directory/path/",
     "lifeDuration": "3600"
   },
   {
     "type": "TIME_BASED",
     "resourceId": "cron",
     "lifeDuration": "0"
   }
 ]
}
```

Time when message is received	Event	Dependency validated	Job triggered
2021-01-01 12:00:00	{ "eventType": "FILE", "eventTimestamp": "2021-01-01 11:59:59", "eventResourceId": "/scheduling_configuraiton_1/directory/path/file_1.txt" }	{ "type": "FILE", "resourceId": "/scheduling_configuraiton_1/directory/path/", "lifeDuration": "3600" }	**No**
2021-01-01 12:05:00	{ "eventType": "FILE", "eventTimestamp": "2021-01-01 12:04:59", "eventResourceId": "/scheduling_configuraiton_1/directory/path/file_2.txt" }	{ "type": "FILE", "resourceId": "/scheduling_configuraiton_1/directory/path/", "lifeDuration": "3600" }	**No**
2021-01-01 12:15:00	{ "eventType": "FILE", "eventTimestamp": "2021-01-01 12:14:50", "eventResourceId": "/scheduling_configuraiton_1/directory/path/file_3.txt" }	{ "type": "FILE", "resourceId": "/scheduling_configuraiton_1/directory/path/", "lifeDuration": "3600" }	**No**
2021-01-01 12:15:30	{ "eventType": "FILE", "eventTimestamp": "2021-01-01 12:15:28", "eventResourceId":	None	**No**

	"/scheduling_configuraiton_1/directory/**an_other_path**/file_3.txt" }		

2021-01-01 12:30:01	{
	"eventType": "TIME_BASED_CRON",
	"eventTimestamp": "2021-01-01 12:30:00",
	"eventResourceId": "cron"
	}
		{
		"type": "TIME_BASED",
		"resourceId": "cron",
		"lifeDuration": "0"
		}
			Yes

2021-01-01 13:10:01	{
	"eventType": "TIME_BASED_CRON",
	"eventTimestamp": "2021-01-01 13:10:00",
	"eventResourceId": "cron"
	}
		{
		"type": "TIME_BASED",
		"resourceId": "cron",
		"lifeDuration": "0"
		}
			Yes

2021-01-01 13:30:01	{
	"eventType": "TIME_BASED_CRON",
	"eventTimestamp": "2021-01-01 13:30:00",
	"eventResourceId": "cron"
	}
		{
		"type": "TIME_BASED",
		"resourceId": "cron",
		"lifeDuration": "0"
		}
			No