

## DP-203 : 16 - Executing ADF Pipelines

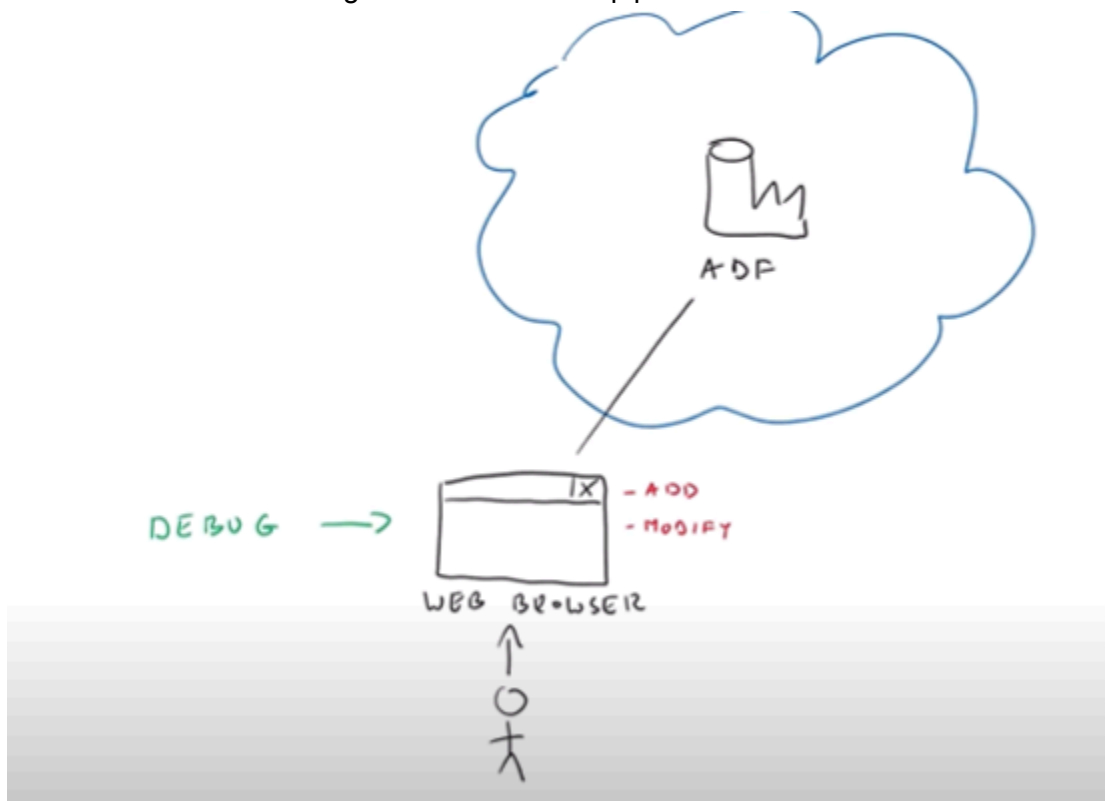
Here we have two options one is DEBUG and other is TRIGGER



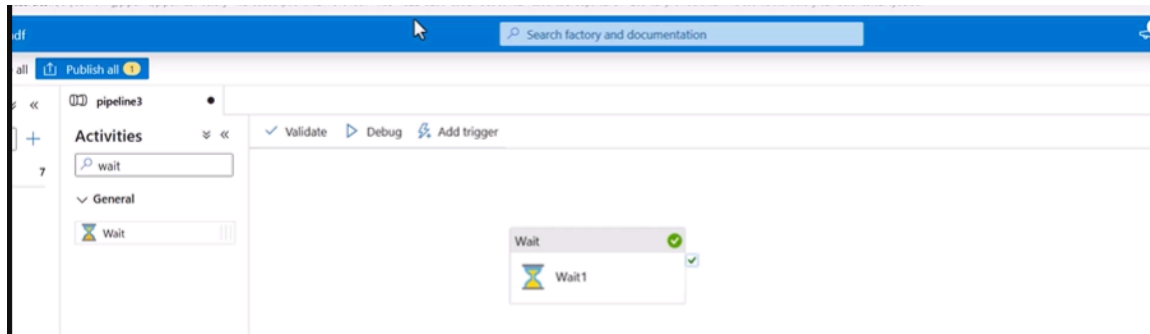
### DEBUG

1. Lets suppose we have ADF in our cloud and the user is accessing the ADF using web interface
2. The user now added some activities to the ADF pipeline and wants to check whether this features are working or not

3. Then he can use the debug mode and test the pipeline

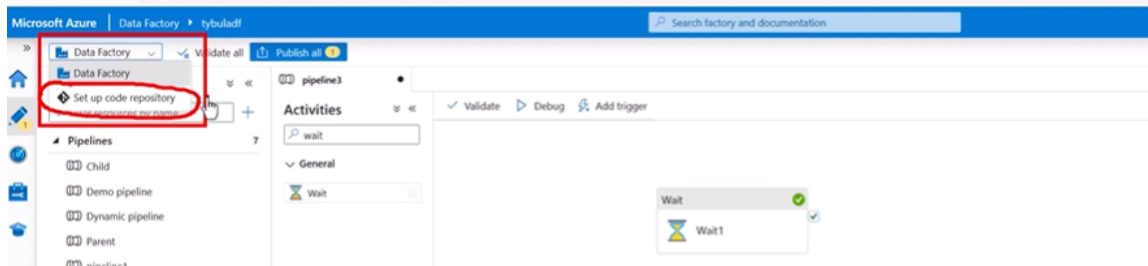


4. Here we have pressed debug and verified our ADF pipeline activities



No Git Integration mode

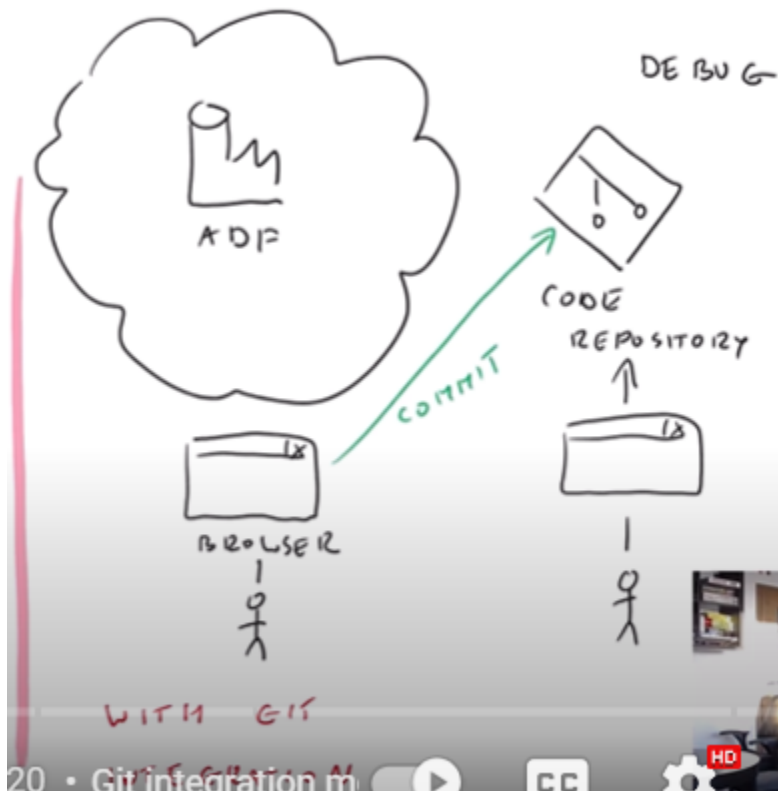
1. ADF can work in 2 modes...with git integration and without



2. Here once we deploy our changes using publish all...we cannot backtrack the changes
3. So main disadvantage is here...we cannot collaborate with others
4. And triggers will execute the changes that are deployed in ADF
5. The main diff bw debug and Trigger is that...in debug it executes the ADF activities loaded to the UI...and triggers will execute the changes that are deployed in ADF

With Git

1. Here if we have git integration enabled



20 • Git integration in here the code repo can be azure DevOps...

2. So here we can save our partial code as commits in the repo...and can come back and work on it later

3. Also here we can collaborate with other developers as well

## Triggers

### Trigger now

1. It is OnDemand Execution which is run manually after we publish our changes in the ADF
2. In the monitor tab...we can check the status of our pipeline in trigger and debug modes

Microsoft Azure | Data Factory | tybuladf

Search factory and documentation

Pipeline runs

Triggered Debug Rerun Cancel options Refresh Edit columns List Gantt

Filter by run ID or name Local time: Last 24 hours Pipeline name: All Status: All Runs: Latest runs Triggered by: All Add filter

Showing 1 - 1 items

| Pipeline name | Run start             | Run end               | Duration | Triggered by   | Status    | Run      |
|---------------|-----------------------|-----------------------|----------|----------------|-----------|----------|
| pipeline3     | 12/1/2023, 8:23:22 AM | 12/1/2023, 8:23:26 AM | 4s       | Manual trigger | Succeeded | Original |

### Scheduled Trigger

1. Here we can run the trigger at particular time every day or every hour

The screenshot shows the 'Schedule trigger' configuration window. At the top, the 'Type' is set to 'Schedule'. The 'Start date' is '12/1/2023, 7:25:24 AM' and the 'Time zone' is 'Sarajevo, Skopje, Warsaw, Zagreb (UTC+1)'. A note indicates that this time zone observes daylight savings. The 'Recurrence' is set to 'Every 1 Day(s)'. Under 'Advanced recurrence options', the 'Execute at these times' section shows 'Hours' set to 1 and 'Minutes' set to 0. The 'Schedule execution times' section shows '01:00'. There is a checkbox for 'Specify an end date' which is currently unchecked. At the bottom, there is an 'Annotations' section with a '+ New' button. A 'Start trigger' button is visible, and a 'Start trigger on creation' checkbox is checked. The interface includes a video player overlay with standard controls like play, pause, and full screen.

Type \*  
Schedule

Start date \* ⓘ  
12/1/2023, 7:25:24 AM

Time zone \* ⓘ  
Sarajevo, Skopje, Warsaw, Zagreb (UTC+1)

ⓘ This time zone observes daylight savings. Trigger will auto-adjust for one hour difference.

Recurrence \* ⓘ  
Every 1 Day(s)

Advanced recurrence options

Execute at these times ⓘ

Hours 1 X

Minutes 0 X

Schedule execution times  
01:00

☐ Specify an end date

Annotations  
+ New

Start trigger ⓘ

☒ Start trigger on creation

2. Here we configured trigger to run on every day at 1:00 am

3. We can also customize them

Type \*

Schedule

Start date \* ⓘ

12/1/2023, 7:25:24 AM

Time zone \* ⓘ

Sarajevo, Skopje, Warsaw, Zagreb (UTC+1)

ⓘ This time zone observes daylight savings. Trigger will auto-adjust for one hour difference.

Recurrence \* ⓘ

Every 1 Week(s)

Advanced recurrence options

Run on these days

|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|-----|-----|-----|-----|-----|-----|-----|

Execute at these times ⓘ

Hours 1 X

Minutes 0 X

Schedule execution times

01:00

☐ Specify an end date

Schedule trigger

like

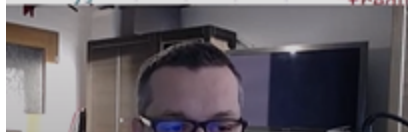
executing every monday and thursday

4. Also we can use this scheduled trigger to use it on multiple pipelines

5. Here we have assigned this trigger to the other pipeline as well

Copy to clipboard

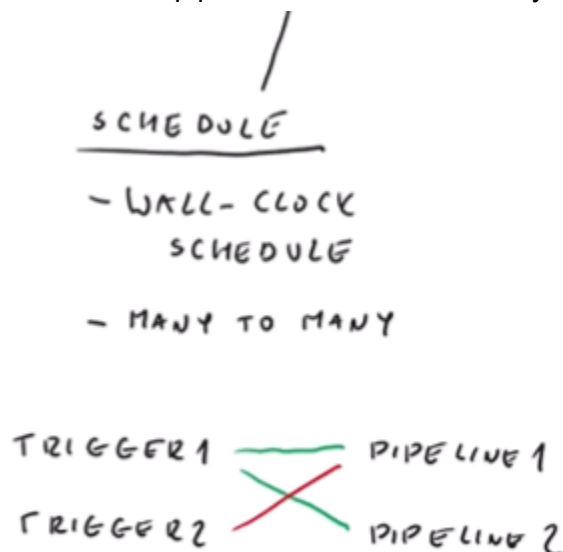
```
1 {
2   "name": "ScheduleTrigger1",
3   "properties": {
4     "annotations": [],
5     "runtimeState": "Started",
6     "pipelines": [
7       {
8         "pipelineReference": {
9           "referenceName": "pipeline3",
10          "type": "PipelineReference"
11        }
12      },
13      {
14        "pipelineReference": {
15          "referenceName": "pipeline4",
16          "type": "PipelineReference"
17        }
18      }
19    ],
20    "type": "ScheduleTrigger",
21    "typeProperties": {
22      "recurrence": {
23        "frequency": "Day",
24        "interval": 1,
25        "startTime": "2023-12-01T07:25:00",
26        "timeZone": "Central European Standard Time",
27        "rule": {
```



we can see this in

the code as well

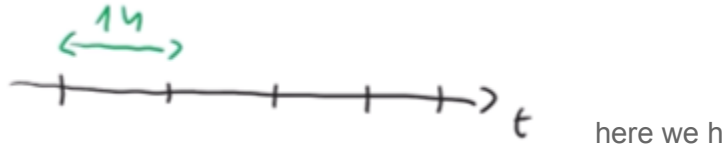
6. And also for one pipeline we can create many schedule triggers...



7. many to many relation schedule trigger and pipelines follow

## Tumbling Window

1. Here the timeline will be splitted into equal intervals



2. Refer video

### Delay:

- The `delay` property specifies a **wait time** after the window ends before the trigger fires and runs the pipeline.
- This is useful for scenarios where the data for the window might not be immediately available after the window closes.

### Example:

Imagine you have a tumbling window trigger set to process sales data for every hour. You suspect there might be a slight delay in data ingestion at the end of each hour. To account for this, you can set a `delay` of 10 minutes in the trigger properties. This ensures the pipeline waits 10 minutes after the hour ends before processing the sales data, allowing enough time for all data to be collected.

### Max Concurrency:

- The `maxConcurrency` property defines the **maximum number of pipeline runs** that can occur simultaneously for different windows.
- This is helpful when dealing with large backfills or situations where running multiple pipelines concurrently might overload resources.

### Example:

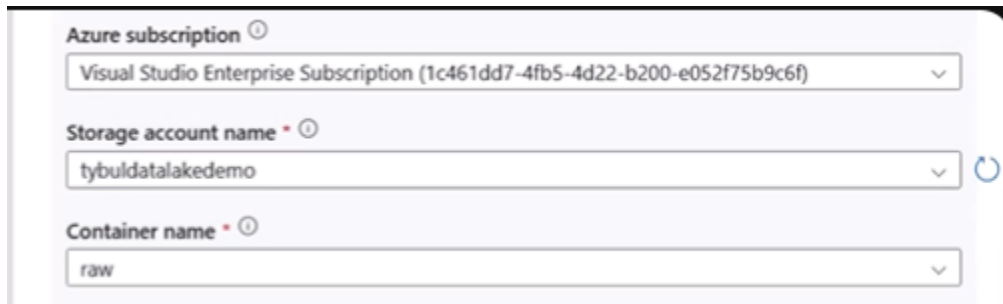
Let's say you're backfilling historical data and want to process data for the last 24 hours using a tumbling window trigger with an hourly window size. Setting `maxConcurrency` to 5 would limit the number of pipeline executions to 5 at any given time. This helps manage resource usage while backfilling data.

## Storage event trigger

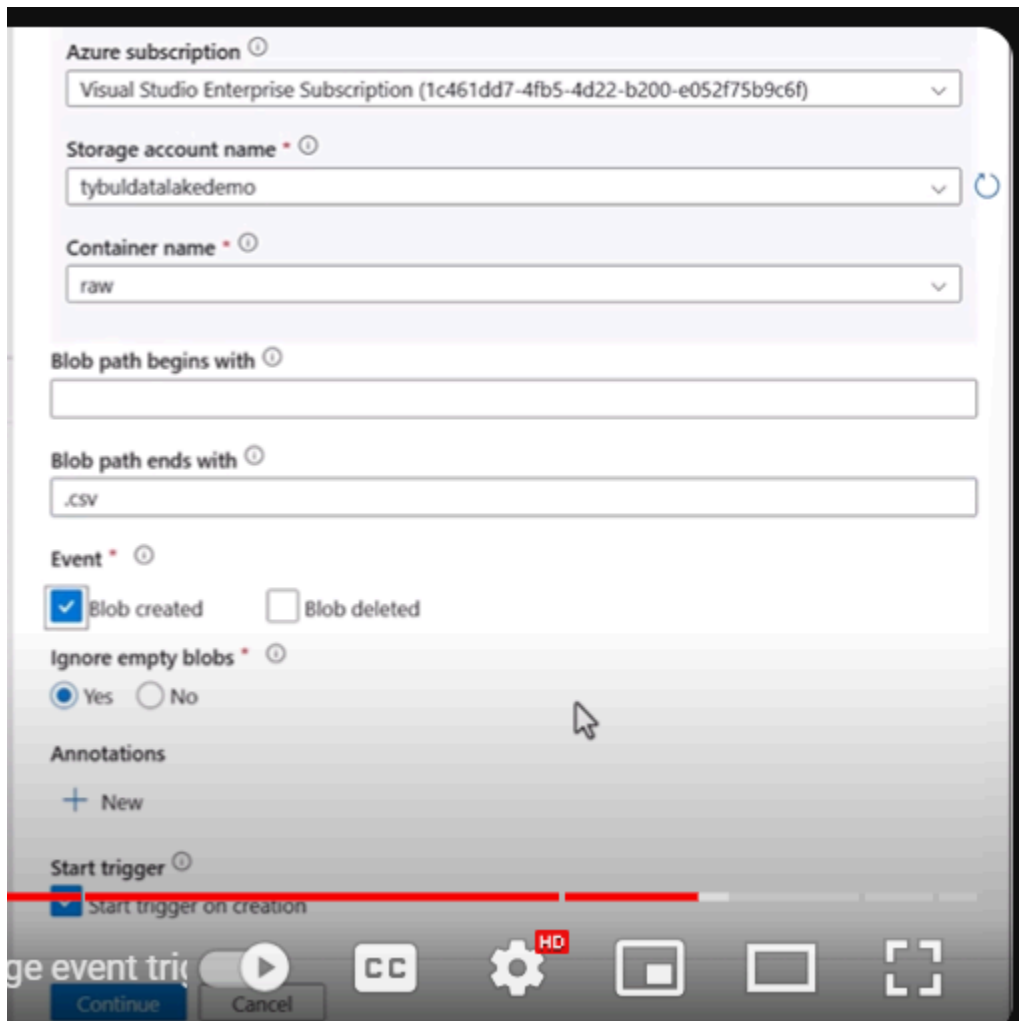
1. We can register our trigger to run...when there is Blob created/deleted



- Here we have to setup the trigger by giving our storage account name(ADL) and the container name



- We can also specify blob's path begin and end



like here  
we have blob end path as “.csv”...so if any csv files gets added or deleted...then our

pipeline will get triggered

The screenshot shows the configuration for a Storage Event Trigger. It includes fields for 'Blob path begins with' and 'Blob path ends with' (containing '.CSV'). Under the 'Event' section, 'Blob created' is selected with a checkbox, while 'Blob deleted' is not. The 'Ignore empty blobs' section has 'Yes' selected. There is a '+ New' button under 'Annotations'. The 'Start trigger' section has 'Start trigger on creation' selected. At the bottom are 'Continue' and 'Cancel' buttons.

## STORAGE EVENT

- BLOB CREATED/DELETED
- EVENT DRIVEN
- USES EVENT GRID

- 4.
5. Storage event trigger...internally uses Event Grid to trigger the pipeline
6. Here event producer would be storage account..



7. The drawback here...is it only works for storage accounts
8. So we have custom event trigger(see in gemini)

