DataFlow Activities Time To Live (TTL) and IR configurations scenarios and recommendations

Last updated by | Ranjith Katukojwala | Jan 31, 2023 at 4:01 PM PST

Contents

- Scenario: 1
- Scenario: 2
- Scenario: 3
- How to check Cluster Compute Acquisition Duration:
- How TTL Works?
    - Possible Questions and Recommendations:
- Additional Information:

By default, every data flow activity spins up a new cluster based on the IR configuration. Cold cluster start-up time takes a few minutes(~2-3 mins) and data processing can't start until it is complete. If most of the data flow executes in parallel, it is not recommended to enable TTL.

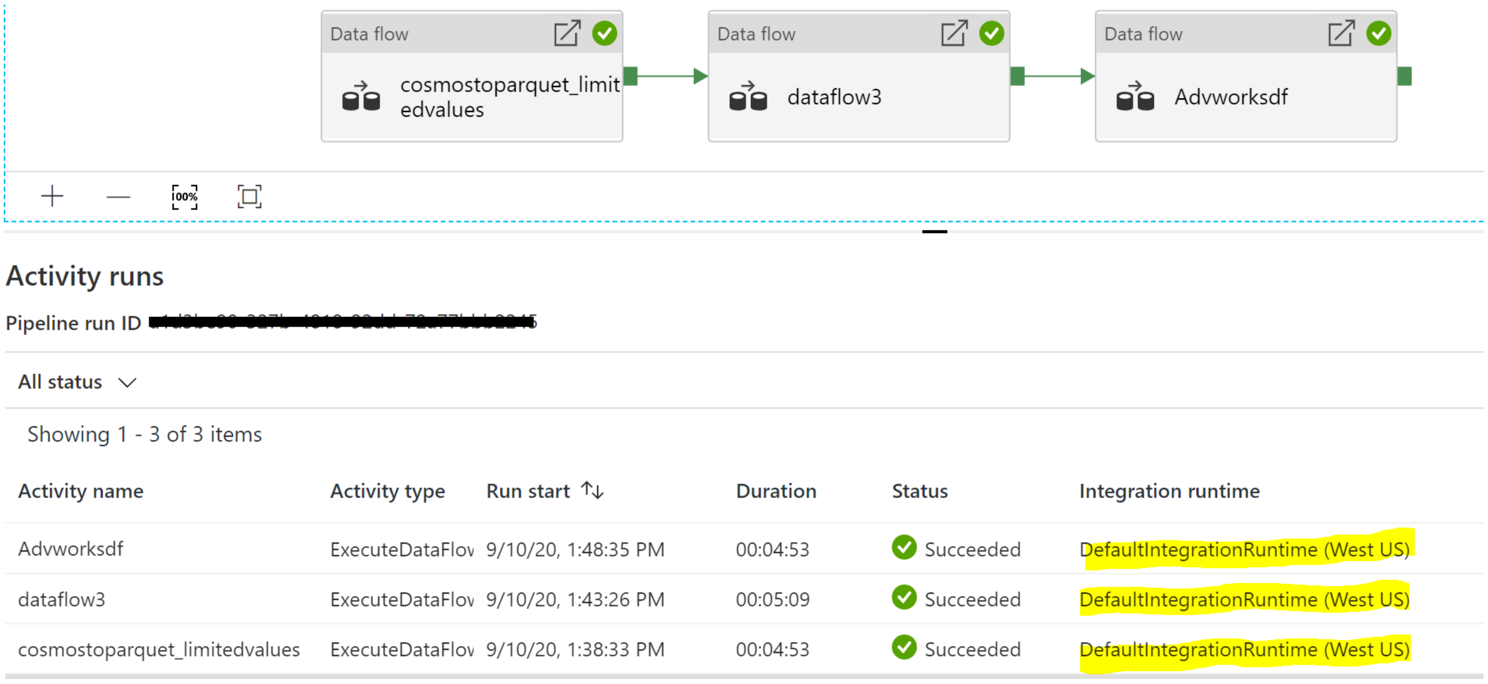
Please note that the Warm cluster takes around ~5-10 sec to pick up the next job.

The customer could set up Dataflow IR TTL in the following Scenarios:

Scenario: 1

Customers could have a pipeline with multiple Dataflow activities with an Integration Runtime (IR) with a Time To Live (TTL) of 0 minutes or DefaultIntegrationRuntime. In this case, a separate cluster will be created for each Dataflow activity and start-up time takes a few minutes for each cluster + processing time.

For instance, the below Pipeline consists of three Dataflow activities and each activity set up with DefaultIntegrationRuntime. In this case, each DF activity will have its own Cluster.



Activity runs

Pipeline run ID [redacted]

All status

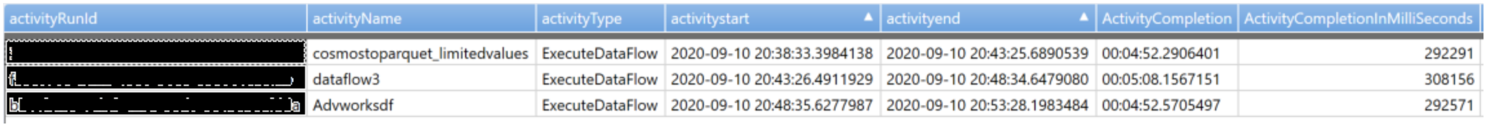
Showing 1 - 3 of 3 items

Activity name	Activity type	Run start	Duration	Status	Integration runtime
Advworksdf	ExecuteDataFlov	9/10/20, 1:48:35 PM	00:04:53	✓ Succeeded	DefaultIntegrationRuntime (West US)
dataflow3	ExecuteDataFlov	9/10/20, 1:43:26 PM	00:05:09	✓ Succeeded	DefaultIntegrationRuntime (West US)
cosmostoparquet_limitedvalues	ExecuteDataFlov	9/10/20, 1:38:33 PM	00:04:53	✓ Succeeded	DefaultIntegrationRuntime (West US)

To check further details:

```
let pipelinerunid = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';
cluster('adfcus.kusto.windows.net').database('AzureDataFactory').ActivityRuns
| union cluster('adfneu.kusto.windows.net').database('AzureDataFactory').ActivityRuns
| where pipelineRunId == pipelinerunid
| where activityType == 'ExecuteDataFlow'
| summarize activitystart = min(start), activityend = max(end),ActivityCompletion = max(end)-min(start), Activ
```

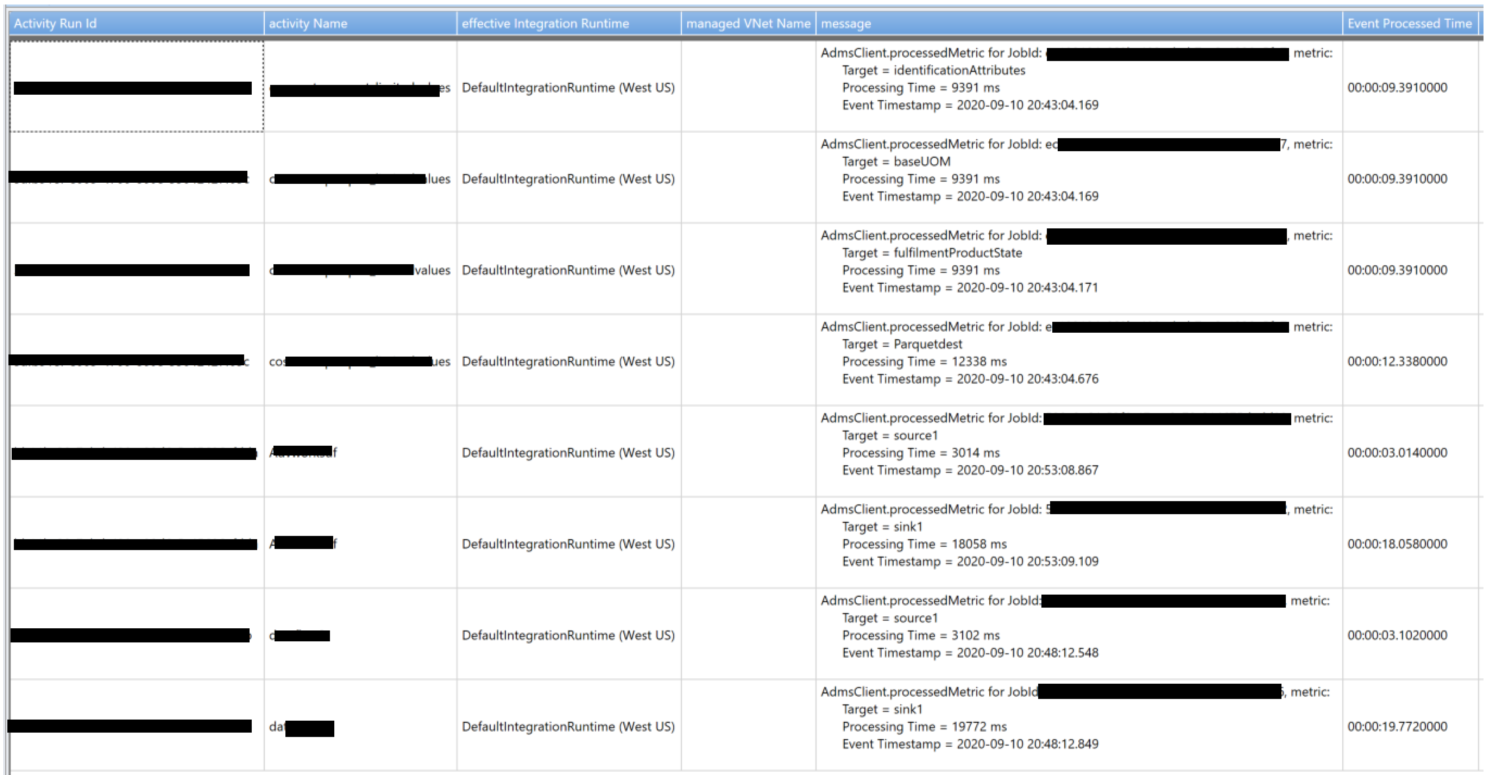
Below result would give the time taken to complete each activity (ActivityCompletion = Cluster Startup time + Processing time)



activityRunId	activityName	activityType	activitystart	activityend	ActivityCompletion	ActivityCompletionInMilliSeconds
[redacted]	cosmostoparquet_limitedvalues	ExecuteDataFlow	2020-09-10 20:38:33.3984138	2020-09-10 20:43:25.6890539	00:04:52.2906401	292291
[redacted]	dataflow3	ExecuteDataFlow	2020-09-10 20:43:26.4911929	2020-09-10 20:48:34.6479080	00:05:08.1567151	308156
[redacted]	Advworksdf	ExecuteDataFlow	2020-09-10 20:48:35.6277987	2020-09-10 20:53:28.1983484	00:04:52.5705497	292571

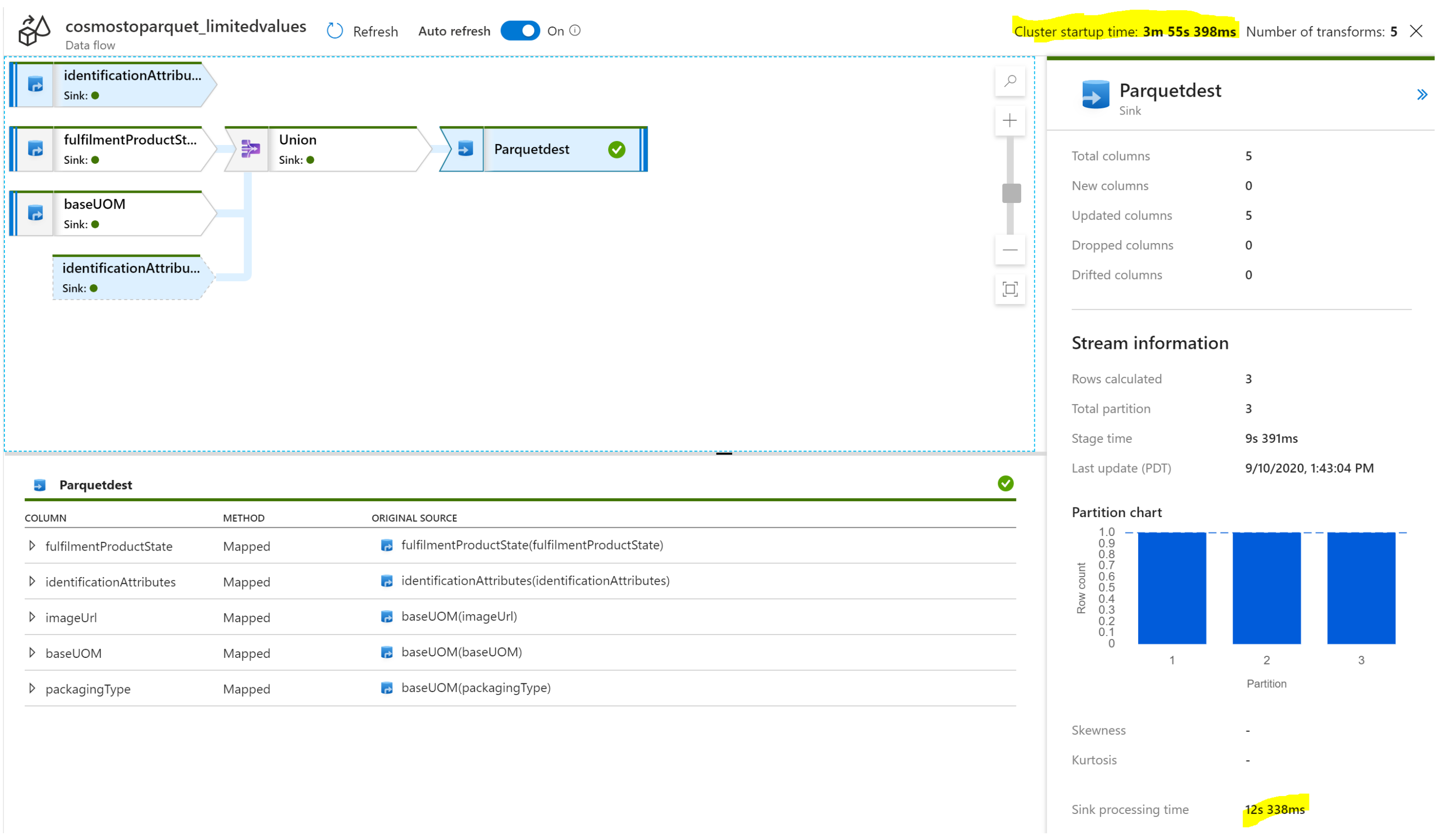
```
let pipelinerunid = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';
cluster('adfcus.kusto.windows.net').database('AzureDataFactory').ActivityRuns
| union cluster('adfneu.kusto.windows.net').database('AzureDataFactory').ActivityRuns
| where pipelineRunId == pipelinerunid
| where activityType == 'ExecuteDataFlow'
| where status != 'Queued' and status != 'InProgress'
| project activityRunId,effectiveIntegrationRuntime,managedVNetName,activityName,pipelineRunId
|join kind= rightouter (
cluster('adfcus.kusto.windows.net').database('AzureDataFactory').DataflowClusterLogs
| union cluster('adfneu.kusto.windows.net').database('AzureDataFactory').DataflowClusterLogs
| where TraceCorrelationId == pipelinerunid
| where Caller == 'com.microsoft.datafactory.dataflow.AdmsClient.processedEvent'
| distinct ActivityRunId, Message,TraceCorrelationId
| extend EventProcessedTime = format_datetime(unixtime_milliseconds_todatetime(toint(extract("Processing Time
| project ActivityRunId,activityName,effectiveIntegrationRuntime,managedVNetName, Message,EventProcessedTime
```

Below result would give the Integration Runtime (IR) used and Time is taken to complete each event in the activities. (EventProcessedTime = Time taken for that event to process).



Activity Run Id	activity Name	effective Integration Runtime	managed VNet Name	message	Event Processed Time
[redacted]	[redacted]es	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId: [redacted] metric: Target = identificationAttributes Processing Time = 9391 ms Event Timestamp = 2020-09-10 20:43:04.169	00:00:09.3910000
[redacted]	d[redacted]lues	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId: ec[redacted]7, metric: Target = baseUOM Processing Time = 9391 ms Event Timestamp = 2020-09-10 20:43:04.169	00:00:09.3910000
[redacted]	d[redacted]values	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId: [redacted], metric: Target = fulfilmentProductState Processing Time = 9391 ms Event Timestamp = 2020-09-10 20:43:04.171	00:00:09.3910000
[redacted]	co[redacted]ues	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId: e[redacted] metric: Target = Parquetdest Processing Time = 12338 ms Event Timestamp = 2020-09-10 20:43:04.676	00:00:12.3380000
[redacted]	A[redacted]f	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId:[redacted] metric: Target = source1 Processing Time = 3014 ms Event Timestamp = 2020-09-10 20:53:08.867	00:00:03.0140000
[redacted]	A[redacted]f	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId: 5[redacted], metric: Target = sink1 Processing Time = 18058 ms Event Timestamp = 2020-09-10 20:53:09.109	00:00:18.0580000
[redacted]	d[redacted]	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId[redacted] metric: Target = source1 Processing Time = 3102 ms Event Timestamp = 2020-09-10 20:48:12.548	00:00:03.1020000
[redacted]	da[redacted]	DefaultIntegrationRuntime (West US)		AdmsClient.processedMetric for JobId[redacted], metric: Target = sink1 Processing Time = 19772 ms Event Timestamp = 2020-09-10 20:48:12.849	00:00:19.7720000

Here Cluster Startup time + Event Processed Time =~ ActivityCompletion. Please check below Activity screenshot for the reference:



cosmostoparquet_limitedvalues
Data flow

Refresh | Auto refresh: On

Cluster startup time: **3m 55s 398ms** Number of transforms: **5**

identificationAttribu... Sink: ●
fulfilmentProductSt... Sink: ●
Union Sink: ●
Parquetdest
baseUOM Sink: ●
identificationAttribu... Sink: ●

Parquetdest

COLUMN	METHOD	ORIGINAL SOURCE
fulfilmentProductState	Mapped	fulfilmentProductState(fulfilmentProductState)
identificationAttributes	Mapped	identificationAttributes(identificationAttributes)
imageUrl	Mapped	baseUOM(imageUrl)
baseUOM	Mapped	baseUOM(baseUOM)
packagingType	Mapped	baseUOM(packagingType)

Parquetdest
Sink

Total columns	5
New columns	0
Updated columns	5
Dropped columns	0
Drifted columns	0

Stream information

Rows calculated	3
Total partition	3
Stage time	9s 391ms
Last update (PDT)	9/10/2020, 1:43:04 PM

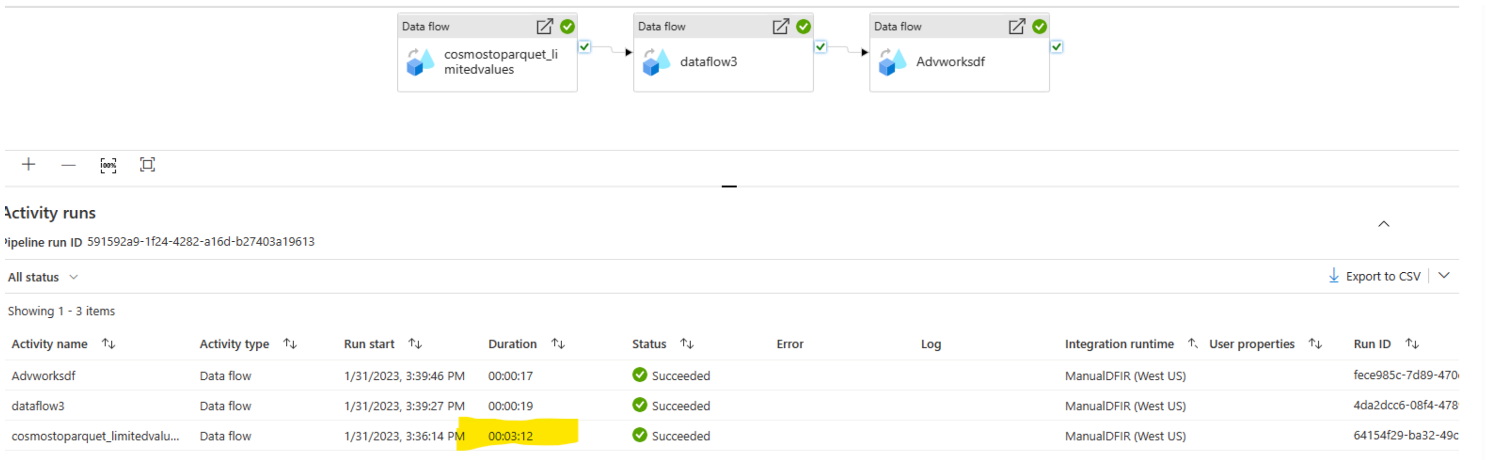
Partition chart

Skewness	-
Kurtosis	-
Sink processing time	12s 338ms

Scenario: 2

Customers could have a pipeline with multiple Dataflow activities with an Integration Runtime (IR) with specified Time To Live (TTL). If the pipelines contain multiple sequential data flow, if we enable a time to live (TTL) value, specifying a time to live value keeps a cluster alive for a certain period of time after its execution completes. If a new job starts using the IR during the TTL time, it will reuse the existing cluster, and start-up time will greatly reduce. After the second job completes, the cluster will again stay alive for the TTL time. Only one job can run on a single cluster at a time. If there is an available cluster, but two data flows start, only one will use the live cluster. The second job will spin up its own isolated cluster.

For instance, the below Pipeline consists of three Dataflow activities with each activity setup with an Azure Integration Runtime with TTL of 10 Minutes. **Note:** The recommendation is to use the smallest TTL, more TTL minutes incur more cost.



Activity runs

Pipeline run ID 591592a9-1f24-4282-a16d-b27403a19613

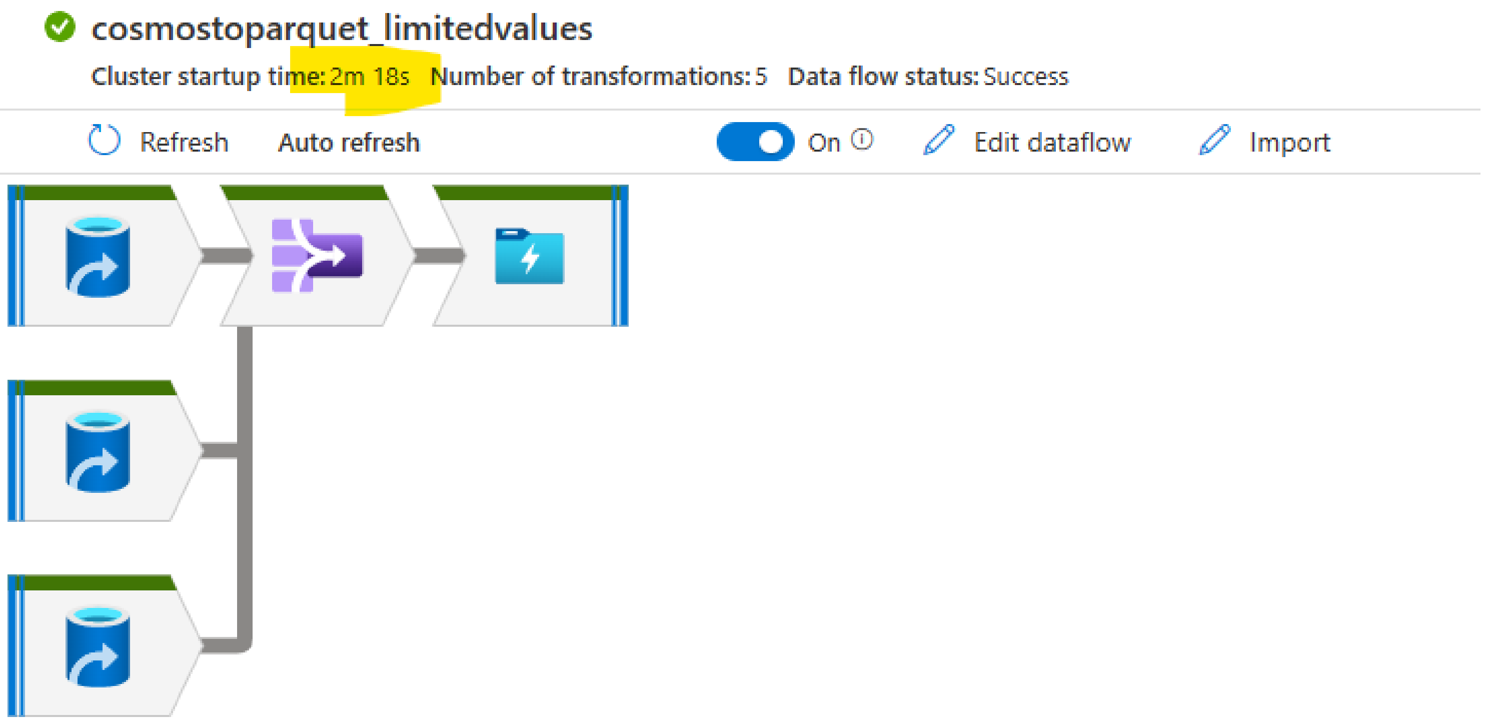
All status

Export to CSV

Showing 1 - 3 items

Activity name	Activity type	Run start	Duration	Status	Error	Log	Integration runtime	User properties	Run ID
Advworksdf	Data flow	1/31/2023, 3:39:46 PM	00:00:17	✓ Succeeded			ManualDFIR (West US)		fece985c-7d89-470
dataflow3	Data flow	1/31/2023, 3:39:27 PM	00:00:19	✓ Succeeded			ManualDFIR (West US)		4da2dcc6-08f4-478
cosmostoparquet_limitedvalu...	Data flow	1/31/2023, 3:36:14 PM	00:03:12	✓ Succeeded			ManualDFIR (West US)		64154f29-ba32-49c

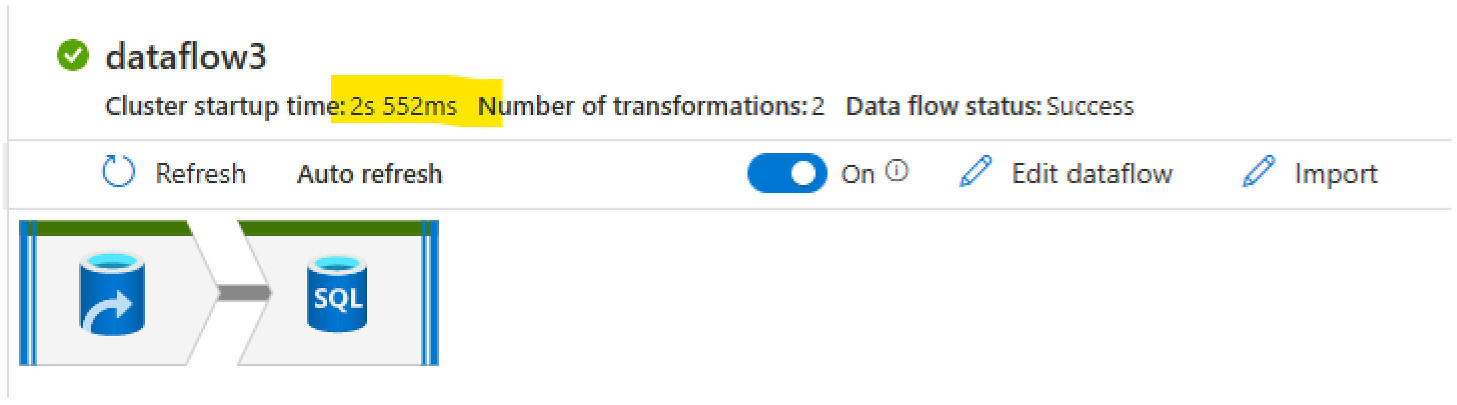
The first Dataflow took around 2m 18s to spin the cold cluster as shown below:


cosmostoparquet_limitedvalues

Cluster startup time: 2m 18s Number of transformations: 5 Data flow status: Success

In this case, although reusing the existing cluster, for second DF activity the dataflow uses warm cluster and it took ~3sec to spin up (shown below). During this time, new containers are being created on that

cluster for isolation, and spark sessions utilize those new isolated containers in the same existing cluster for job processing. ADF - Dataflow keeps all jobs isolated from each other. When we submit a job they create a new container and start spark again for isolation. So you won't find that operation in ADF logs. Customers could use this Scenario to enhance performance.



dataflow3

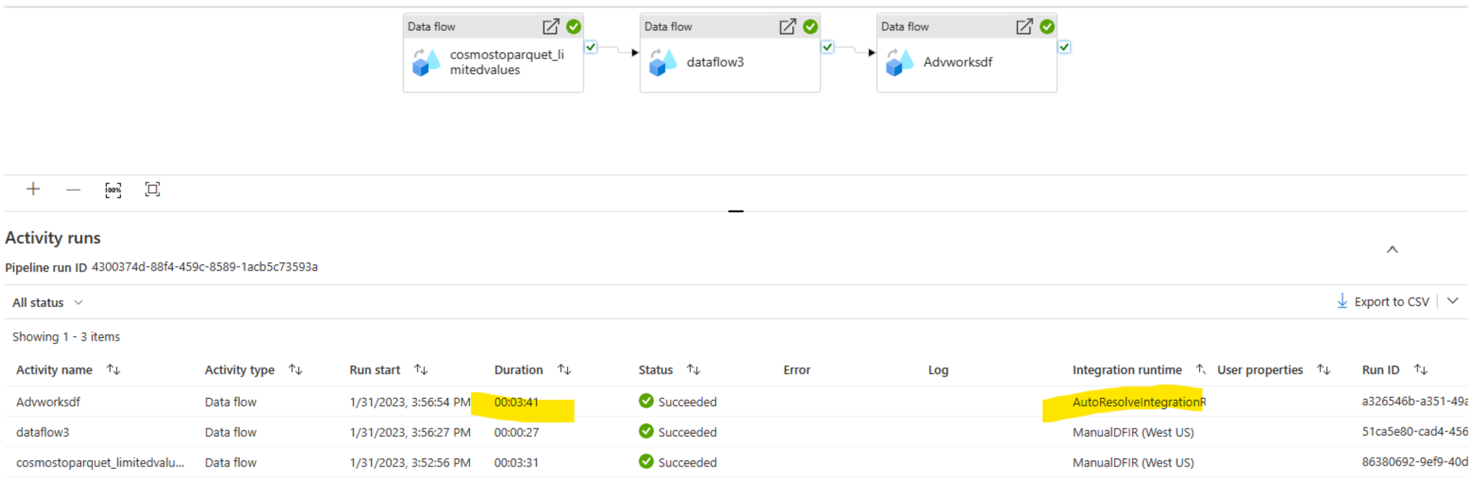
Cluster startup time: 2s 552ms Number of transformations: 2 Data flow status: Success

We can use the above Kusto query to find the telemetry of detailed timelines of activities.

Scenario: 3

Customers could have a pipeline with multiple Dataflow activities with an Integration Runtime (IR) with specified Time To Live (TTL) and default Integration Runtime set up one for each Dataflow Activity.

For instance, the following Pipeline consists of three Dataflow activities. Out of all activities, 2 DF activities setup with an Azure Integration Runtime with TTL of 10 Minutes while the remaining one activity setup with Default Integration Runtime.



Activity runs

Pipeline run ID 4300374d-88f4-459c-8589-1acb5c73593a

All status

Showing 1 - 3 items

Activity name	Activity type	Run start	Duration	Status	Error	Log	Integration runtime	User properties	Run ID
Advworksdf	Data flow	1/31/2023, 3:56:54 PM	00:03:41	✓ Succeeded			AutoResolveIntegrationR		a326546b-a351-49a
dataflow3	Data flow	1/31/2023, 3:56:27 PM	00:00:27	✓ Succeeded			ManualDFIR (West US)		51ca5e80-cad4-456
cosmostoparquet_limitedvalu...	Data flow	1/31/2023, 3:52:56 PM	00:03:31	✓ Succeeded			ManualDFIR (West US)		86380692-9ef9-40d

We can use the above Kusto query to find the telemetry of the completion time.

How to check Cluster Compute Acquisition Duration:

Please refer to this [wiki]()

How TTL Works?

Please refer to this wiki

Possible Questions and Recommendations:

1. What exactly happens during Cluster startup time besides cluster creation; warmup? Do we submit the job to the cluster? **Ans: See explanation in Scenario 1 and Scenario 2**
2. Can cluster startup time can be reduced/avoided in Scenario 2? **Ans: No, see explanation in Scenario 2**

Additional Information:

How good have you found this content?