WEBVTT

68713812-3367-408c-94e8-c1c0fc9881ad-0
00:00:38.000 --> 00:00:38.720
Hey, everyone.

92be8bcc-4c14-4c16-9843-123079da0616-0
00:00:40.360 --> 00:00:40.920
Hi, Siddharth.

6bcbd7ae-f5c9-497b-8eb1-b85780742d4c-0
00:00:41.280 --> 00:00:41.800
Hi everybody.

9724f757-1e59-4dcb-ac2a-e90187298a87-0
00:00:43.040 --> 00:00:50.371
Yes, I see most probably the
last session, I mean the last

9724f757-1e59-4dcb-ac2a-e90187298a87-1
00:00:50.371 --> 00:00:52.360
service session.

33384bbc-f873-4905-9112-e6fc512778fa-0
00:00:52.440 --> 00:00:56.080
So we might have one more for
the assessment as well.

2b7315c8-b493-4eb1-bdd2-c7d09dd417ef-0
00:01:02.160 --> 00:01:02.280
OK.

168a5331-cfaa-488f-a0a6-13b13e392010-0
00:01:02.280 --> 00:01:04.240
So let's give a couple of
minutes for this to join.

22e1f63c-3840-4e71-bfaa-38eddd7913b8-0
00:01:15.050 --> 00:01:15.610
OK.

2a8b1412-6468-4f83-81fa-e8eded8c669b-0
00:01:22.430 --> 00:01:26.718
So I think we had total of 14
sessions and this is the 15th,

2a8b1412-6468-4f83-81fa-e8eded8c669b-1
00:01:26.718 --> 00:01:27.070
1:00.

f78dd329-a6e5-4d4a-9a3f-e8d322a6abc8-0
00:01:29.750 --> 00:01:32.110
Yeah, that's right.

37755d42-f07b-45c5-916d-2a08a5df47a7-0
00:01:33.040 --> 00:01:35.600
So today we'll be talking about
batch processing.

efb2b6bb-59cd-4cb1-9af2-97436a10c66c-0
00:01:35.840 --> 00:01:40.910
And again, similar to error
handling, batch processing has a

efb2b6bb-59cd-4cb1-9af2-97436a10c66c-1

00:01:40.910 --> 00:01:42.240
lot of concepts.

7c30477a-83d2-478b-8641-d765ca7988e1-0
00:01:42.240 --> 00:01:45.473
So that's the reason I wanted to
have, you know, extra 30

7c30477a-83d2-478b-8641-d765ca7988e1-1
00:01:45.473 --> 00:01:45.920
minutes.

7fc5a9cc-35ec-4591-860e-764699a18b58-0
00:01:46.800 --> 00:01:48.840
I hope it's fine with everyone.

ffb02a6c-b256-450d-9224-e0b556092413-0
00:01:48.840 --> 00:01:52.280
You know, we'll be extending,
you know, by 30 minutes.

af0e1db4-8dd5-4c1f-a8e1-da1cbfe8a346-0
00:01:57.640 --> 00:01:58.800
Yeah, I think that should be OK.

d04be9ef-9443-429e-8b34-0585c4a1b536-0
00:02:00.400 --> 00:02:00.720
OK.

8343cbe5-4b79-46d6-9c84-6cc15b559b39-0
00:02:00.800 --> 00:02:01.000
Yeah.

4440de7e-7400-44c3-b04c-5a7daba7eed6-0
00:02:01.040 --> 00:02:01.440
Thanks.

65b7fa64-aa2b-4a18-8bc5-3077daf57ed9-0
00:02:03.040 --> 00:02:04.520
So let me share my screen.

71655f99-5a07-4336-8980-2e392cc9e9ec-0
00:02:05.240 --> 00:02:08.571
I'll wait for two more minutes
because I don't want others to,

71655f99-5a07-4336-8980-2e392cc9e9ec-1
00:02:08.571 --> 00:02:11.902
you know, miss the starting 5
minutes because they're going to

71655f99-5a07-4336-8980-2e392cc9e9ec-2
00:02:11.902 --> 00:02:12.960
miss the core thing.

7db60125-d305-4c30-8cbe-58bce9d57f1c-0
00:02:12.960 --> 00:02:16.302
So in the meantime, please, you
know, if you, if you can you

7db60125-d305-4c30-8cbe-58bce9d57f1c-1
00:02:16.302 --> 00:02:19.480
know, ask your team members to
join, that would be great.

a10094ad-04b8-4548-9d02-5f3af6164262-0
00:02:33.820 --> 00:02:38.569
And since we are waiting for
people to join, so talking

a10094ad-04b8-4548-9d02-5f3af6164262-1
00:02:38.569 --> 00:02:43.489
about, so talking about what we
say the e-learning module

a10094ad-04b8-4548-9d02-5f3af6164262-2
00:02:43.489 --> 00:02:48.493
status, I think four to five
people have completed all the

a10094ad-04b8-4548-9d02-5f3af6164262-3
00:02:48.493 --> 00:02:53.921
modules which does not include
Divya Tej because he has already

a10094ad-04b8-4548-9d02-5f3af6164262-4
00:02:53.921 --> 00:02:54.600
done it.

dce6f997-9aab-4bf1-9826-5aa5f0a10d52-0
00:02:55.120 --> 00:03:00.240
And a few people are in 13th
module, few are in 9 or 8.

62150ff0-a5a2-4d46-8b73-bb1ba40f314a-0
00:03:00.680 --> 00:03:04.104
So please make sure you try to
complete all the modules by the

62150ff0-a5a2-4d46-8b73-bb1ba40f314a-1
00:03:04.104 --> 00:03:04.920
coming Tuesday.

9a3fef05-734b-4efe-838e-137fbe92942c-0
00:03:05.040 --> 00:03:09.446
OK, so when I say Tuesday it is
19th because on 19th, you know I

9a3fef05-734b-4efe-838e-137fbe92942c-1
00:03:09.446 --> 00:03:13.040
would like to roll out the
second hands on exercise.

2c5b1af0-d228-4181-b3a1-261445f9a0a3-0
00:03:13.400 --> 00:03:16.964
So in that exercise it covers
all the modules up to batch

2c5b1af0-d228-4181-b3a1-261445f9a0a3-1
00:03:16.964 --> 00:03:17.640
processing.

9488a170-46ed-4664-a2fe-363d455ddb3f-0
00:03:17.840 --> 00:03:20.200
So please try to complete it by
Tuesday.

0905eb34-3495-476e-9feb-857e0bdc455f-0
00:03:21.800 --> 00:03:22.040
OK.

475d763a-287d-438c-83a6-b6c8fdd1c13c-0
00:03:22.040 --> 00:03:26.920
I think even we have crossed
almost 7 to 9 weeks, right?

9703a5d5-bfe5-4f4a-b9bb-d7e83ddfc705-0
00:03:26.920 --> 00:03:30.414
We started this around 8 to 9
weeks ago, so please, please try

9703a5d5-bfe5-4f4a-b9bb-d7e83ddfc705-1
00:03:30.414 --> 00:03:33.686
to, you know, work on the
weekends and not try to complete

9703a5d5-bfe5-4f4a-b9bb-d7e83ddfc705-2
00:03:33.686 --> 00:03:35.240
as much as possible by 19th.

6b41dcad-2176-4927-ade1-7414674b2a37-0
00:03:38.080 --> 00:03:38.360
OK.

ae51dc73-f455-477d-b684-c617507c0a20-0
00:03:38.360 --> 00:03:41.963
So with that said, so we only
have 15 people, but I'll go

ae51dc73-f455-477d-b684-c617507c0a20-1
00:03:41.963 --> 00:03:44.200
ahead because we need all the
time.

2720f7bf-5b0a-494d-bb05-5546693ce836-0
00:03:45.680 --> 00:03:46.720
So can you just see my screen?

cf11a795-27eb-45f7-b38b-e36987d2d458-0
00:03:50.480 --> 00:03:53.480
Yes, if that we could able to
see your screen, that's great.

e4471963-e418-4a16-a9da-0b22df6d04b6-0
00:03:54.000 --> 00:03:57.335
So today we'll be talking about
batch processing and in batch

e4471963-e418-4a16-a9da-0b22df6d04b6-1
00:03:57.335 --> 00:04:00.241
processing these are the
different topics or concepts

e4471963-e418-4a16-a9da-0b22df6d04b6-2
00:04:00.241 --> 00:04:01.639
which we will be covering.

573fc64f-fbf4-4752-be3b-f6af6377e64b-0
00:04:01.640 --> 00:04:06.072
So it is again very tricky part
and even in the fundamentals

573fc64f-fbf4-4752-be3b-f6af6377e64b-1
00:04:06.072 --> 00:04:10.796
training we just cover you know

a few of the concepts not all of

573fc64f-fbf4-4752-be3b-f6af6377e64b-2
00:04:10.796 --> 00:04:11.160
them.

9631a407-06f8-41e1-b1a4-acecc328df09-0
00:04:11.320 --> 00:04:14.907
So we'll, you know, start with
the introduction to batch

9631a407-06f8-41e1-b1a4-acecc328df09-1
00:04:14.907 --> 00:04:15.600
processing.

8c1575f7-5b45-4b32-b74a-99b2462d28fd-0
00:04:15.720 --> 00:04:19.448
I have I think 3 to 4 slides
which talks about what is batch,

8c1575f7-5b45-4b32-b74a-99b2462d28fd-1
00:04:19.448 --> 00:04:23.177
why you need, why you need to
use it and how batch works, you

8c1575f7-5b45-4b32-b74a-99b2462d28fd-2
00:04:23.177 --> 00:04:24.199
know, internally.

dc4377ba-b034-440f-b102-e8b5b4cf75d1-0
00:04:24.880 --> 00:04:26.400
And then we'll do a demo.

f43d5ab4-7305-4c75-8808-d855b1282e19-0
00:04:26.400 --> 00:04:29.040
So we'll we'll do a demo with
two sets of records.

f248c197-9557-4fd4-b58c-fbec584072cc-0
00:04:29.040 --> 00:04:32.910
So one we'll be using just five
records to understand you know

f248c197-9557-4fd4-b58c-fbec584072cc-1
00:04:32.910 --> 00:04:34.200
most of the concepts.

86eaf8ae-a361-4a85-9872-81d90402fd86-0
00:04:34.600 --> 00:04:39.026
And then we'll also use 1000
record set to understand us in a

86eaf8ae-a361-4a85-9872-81d90402fd86-1
00:04:39.026 --> 00:04:40.240
specific concept.

66da1b74-1484-483f-9e3e-4a8f77fe361b-0
00:04:40.920 --> 00:04:43.560
OK, so let's go ahead.

fad4c1c2-36b3-4d78-8a7b-081ce2685d1f-0
00:04:44.240 --> 00:04:47.102
Ideally speaking, batch
processing is used where you

fad4c1c2-36b3-4d78-8a7b-081ce2685d1f-1
00:04:47.102 --> 00:04:49.640
want to, you know process
millions of records.

7f26cfab-aed9-43df-93ea-2cee71ca231c-0
00:04:49.640 --> 00:04:51.600
It's not used for a small set of
records.

ce1077bb-8ccc-4256-a842-f1b8a9c7205f-0
00:04:51.600 --> 00:04:54.482
So if you want to process
hundreds of thousands of records

ce1077bb-8ccc-4256-a842-f1b8a9c7205f-1
00:04:54.482 --> 00:04:57.413
or millions of records, you
know, you then make use of, you

ce1077bb-8ccc-4256-a842-f1b8a9c7205f-2
00:04:57.413 --> 00:04:57.999
know, batch.

d03d4b33-65e1-4dc8-b07c-e8e7fb0a3d37-0
00:04:58.040 --> 00:05:01.587
Because batch is designed for
reliability, it does

d03d4b33-65e1-4dc8-b07c-e8e7fb0a3d37-1
00:05:01.587 --> 00:05:05.621
asynchronous processing and it
can be used for data which

d03d4b33-65e1-4dc8-b07c-e8e7fb0a3d37-2
00:05:05.621 --> 00:05:07.360
cannot be hold in memory.

e4dd826e-a1e3-4682-8d12-0094b851f0f3-0
00:05:07.360 --> 00:05:10.310
So if you're talking about
million records, you cannot hold

e4dd826e-a1e3-4682-8d12-0094b851f0f3-1
00:05:10.310 --> 00:05:12.080
million records in a memory,
right?

1da8e28f-89c4-4d02-97ed-cd540dbab371-0
00:05:12.080 --> 00:05:15.850
Because the maximum if you go
with cloud up 1.0 or the maximum

1da8e28f-89c4-4d02-97ed-cd540dbab371-1
00:05:15.850 --> 00:05:19.380
RAM size is around 32GB and if
you're trying to store huge

1da8e28f-89c4-4d02-97ed-cd540dbab371-2
00:05:19.380 --> 00:05:23.210
records it it will give you out
of memory errors so right so so

1da8e28f-89c4-4d02-97ed-cd540dbab371-3
00:05:23.210 --> 00:05:26.621

those scenarios would are the
best cases for using batch

1da8e28f-89c4-4d02-97ed-cd540dbab371-4
00:05:26.621 --> 00:05:27.280
processing.

14324f8e-62c6-4684-a70f-cd915d47f50c-0
00:05:28.280 --> 00:05:32.240
So what batch actually does is
when I said reliability it's

14324f8e-62c6-4684-a70f-cd915d47f50c-1
00:05:32.240 --> 00:05:36.597
going to automatically split the
input data and store them into a

14324f8e-62c6-4684-a70f-cd915d47f50c-2
00:05:36.597 --> 00:05:37.720
persistent queue.

20e4b559-d977-4b28-b27d-a2be07f3af9f-0
00:05:38.120 --> 00:05:41.687
So the reason why it is using
persistent queue is even when

20e4b559-d977-4b28-b27d-a2be07f3af9f-1
00:05:41.687 --> 00:05:45.314
the application crashes or if
the application gets restarted

20e4b559-d977-4b28-b27d-a2be07f3af9f-2
00:05:45.314 --> 00:05:49.238
the data is not lost and the job
execution it was going to resume

20e4b559-d977-4b28-b27d-a2be07f3af9f-3
00:05:49.238 --> 00:05:51.320
from the point it stopped
earlier.

c028250c-243c-416c-97b8-c6783ef8a11c-0
00:05:51.640 --> 00:05:54.491
OK, so that is possible in batch
and in the last session we

c028250c-243c-416c-97b8-c6783ef8a11c-1
00:05:54.491 --> 00:05:55.680
talked about Farid scope.

edfed4ec-5f8b-44b1-85b4-0e0026ebef28-0
00:05:55.680 --> 00:06:00.380
So Farid scope one it is not
asynchronous, it uses a single

edfed4ec-5f8b-44b1-85b4-0e0026ebef28-1
00:06:00.380 --> 00:06:03.200
thread and it is not that
reliable.

f6b57229-13fc-4c27-9428-8b28ba799858-0
00:06:03.200 --> 00:06:06.464
If the application crashes, the
Farid scope should start from

f6b57229-13fc-4c27-9428-8b28ba799858-1
00:06:06.464 --> 00:06:07.360
beginning itself.

175cd958-a9e7-45a9-937a-6f34e0d54d09-0
00:06:07.960 --> 00:06:11.682
OK And some of the common use
cases for batch processing you

175cd958-a9e7-45a9-937a-6f34e0d54d09-1
00:06:11.682 --> 00:06:15.466
know includes if you want to
synchronize the data between two

175cd958-a9e7-45a9-937a-6f34e0d54d09-2
00:06:15.466 --> 00:06:18.640
different business applications,
you can use badge.

93ea9d70-66ad-4830-b34e-9533484a1c27-0
00:06:18.640 --> 00:06:21.665
If you want to do ETL like
extracting, transforming,

93ea9d70-66ad-4830-b34e-9533484a1c27-1
00:06:21.665 --> 00:06:24.520
loading the data such As for
millions of records.

2640ca2d-808f-4684-8f59-1e5f3c020c89-0
00:06:24.640 --> 00:06:26.440
Again you can make use of batch.

02595a3f-316d-4d40-ae54-a5a087b39df6-0
00:06:26.960 --> 00:06:30.880
And if you see this diagram this
is how a bad scope looks like.

b0a67887-20cb-4bda-8778-54c4ab3ae51e-0
00:06:31.280 --> 00:06:35.682
The bad scope is added within a
you know a mule flow within any

b0a67887-20cb-4bda-8778-54c4ab3ae51e-1
00:06:35.682 --> 00:06:40.084
regular flow you have a source,
you have error handling and you

b0a67887-20cb-4bda-8778-54c4ab3ae51e-2
00:06:40.084 --> 00:06:44.280
have this bad scope similar to
cache scope Tri scope, right?

90fc983c-8945-4860-b074-f0806b7c5c3a-0
00:06:44.280 --> 00:06:47.510
So you have a bad scope within
the bad scope, so you have

90fc983c-8945-4860-b074-f0806b7c5c3a-1
00:06:47.510 --> 00:06:48.680
multiple batch steps.

8210064b-2237-444e-8201-af908a12dda6-0
00:06:48.960 --> 00:06:52.320
Each batch step has some

processing phase, some

8210064b-2237-444e-8201-af908a12dda6-1
00:06:52.320 --> 00:06:54.560
aggregator and some on complete.

e77fe263-f1f4-4a78-a8f9-271ee7d4e95d-0
00:06:54.720 --> 00:06:56.440
So we will talk about all these
things.

b1bf6469-eeed-4b86-8c4a-e9c3509ea9f1-0
00:06:56.680 --> 00:06:58.960
OK, so let me move forward.

39a563a4-5e33-49e5-8f2c-8d3dc4ab6edb-0
00:06:59.480 --> 00:07:03.120
So there are three different
phases in a batch job.

d71e9265-3817-4db5-9ad1-b4374183c5e3-0
00:07:03.360 --> 00:07:07.480
So these are load and dispatch
process and on complete.

1c658b06-b987-44df-97f2-ac36834547e6-0
00:07:07.800 --> 00:07:11.276
So if you see the diagram here,
in the batch job you could see

1c658b06-b987-44df-97f2-ac36834547e6-1
00:07:11.276 --> 00:07:13.760
two phases, process records and
on complete.

90c54ec5-adae-4e9a-8af5-a6b074723d98-0
00:07:13.760 --> 00:07:15.720
So these are the 2 phases out of
the three.

bb902532-a5a9-4532-8636-f3d0831b18ca-0
00:07:16.120 --> 00:07:17.320
But what about the first one?

39b51a00-2a38-435e-b68d-fac424033552-0
00:07:17.680 --> 00:07:19.440
So the load and dispatch phase?

016421d0-1404-4d75-ae89-66f29de80ff8-0
00:07:19.560 --> 00:07:24.330
It is responsible for splitting
the input payload into records

016421d0-1404-4d75-ae89-66f29de80ff8-1
00:07:24.330 --> 00:07:29.100
and this will be persisted in a
persistent queue and this load

016421d0-1404-4d75-ae89-66f29de80ff8-2
00:07:29.100 --> 00:07:32.280
and dispatch phase happens
automatically.

158a406d-bdb5-48b7-a0fc-64cab97c287a-0
00:07:32.280 --> 00:07:34.480

So as a developer you don't need
to do anything.

0b4bad4f-6145-47de-8af6-8f8f38808b80-0
00:07:34.640 --> 00:07:38.387
When the batch job receives
those millions of records, it is

0b4bad4f-6145-47de-8af6-8f8f38808b80-1
00:07:38.387 --> 00:07:41.458
internally going to
automatically split them into

0b4bad4f-6145-47de-8af6-8f8f38808b80-2
00:07:41.458 --> 00:07:44.960
individual records and place
them in a persistent queue.

eda78bcc-0dce-4784-acdd-3f055ec952c7-0
00:07:45.080 --> 00:07:47.040
So that is the implicit phase.

6b8f3206-fb44-4bbd-9cbe-1ba1a59d8be1-0
00:07:47.040 --> 00:07:50.078
So the load and dispatch is a
implicit phase which happens

6b8f3206-fb44-4bbd-9cbe-1ba1a59d8be1-1
00:07:50.078 --> 00:07:50.800
automatically.

18bca83f-8246-4fc1-a611-1644fae164e6-0
00:07:51.240 --> 00:07:54.320
So once so this is how a queue
will be created.

fcb604ea-f8d7-4e6c-b366-46bb42a71262-0
00:07:54.320 --> 00:07:58.086
OK so a persistent VM queue will
be created and all the records

fcb604ea-f8d7-4e6c-b366-46bb42a71262-1
00:07:58.086 --> 00:08:01.912
will be stored in that and each
record also has something called

fcb604ea-f8d7-4e6c-b366-46bb42a71262-2
00:08:01.912 --> 00:08:03.560
SMM is nothing but metadata.

6792e93e-7bfb-4bde-83e3-b38979383071-0
00:08:03.560 --> 00:08:05.360
So I'll explain what is the
reason for this.

af3179c0-ab04-411e-a561-53bea532021f-0
00:08:05.360 --> 00:08:09.129
So so if you take example of
1000 records, so all the 1000

af3179c0-ab04-411e-a561-53bea532021f-1
00:08:09.129 --> 00:08:12.835
records will be you know
splitted into individual records

af3179c0-ab04-411e-a561-53bea532021f-2
00:08:12.835 --> 00:08:15.199
and this will be persisted in a
VMQ.

d4f047e1-8da0-4092-b6c8-1de6ea36423c-0
00:08:15.200 --> 00:08:18.040
So this is done using the load
and dispatch phase.

2e25cbc6-da61-454b-ab35-cce4849ca4a1-0
00:08:18.520 --> 00:08:20.840
So then the process phase
begins.

4b3708f7-57d5-49e9-8fa4-e4ffe271e62d-0
00:08:21.000 --> 00:08:25.704
So in the process phase you make
use of one or multiple batch

4b3708f7-57d5-49e9-8fa4-e4ffe271e62d-1
00:08:25.704 --> 00:08:26.160
steps.

a34412dd-90e6-46ff-b2fa-1df6ea2639f3-0
00:08:26.520 --> 00:08:30.579
So in each batch step you can do
a specific, you know, business

a34412dd-90e6-46ff-b2fa-1df6ea2639f3-1
00:08:30.579 --> 00:08:30.960
logic.

e6c2fe99-9d6b-4bde-bcdd-d3ff5106bf83-0
00:08:30.960 --> 00:08:32.760
So let's say you want to do some
ETL.

70fb63f1-ddf7-4299-ab57-718e032e10ba-0
00:08:32.960 --> 00:08:36.609
So in one batch step you can
apply some logic and extract the

70fb63f1-ddf7-4299-ab57-718e032e10ba-1
00:08:36.609 --> 00:08:37.080
records.

a8760a8c-b2d3-4086-b9e1-e752e63fb0e0-0
00:08:37.280 --> 00:08:40.612
In another batch step you can
transform the data as required

a8760a8c-b2d3-4086-b9e1-e752e63fb0e0-1
00:08:40.612 --> 00:08:44.053
and in another batch step you
can load the data to some target

a8760a8c-b2d3-4086-b9e1-e752e63fb0e0-2
00:08:44.053 --> 00:08:46.840
system, let's say from SQL
database to Salesforce.

a7a9f09c-6b50-4587-b422-97574cf6149c-0
00:08:47.120 --> 00:08:47.400
OK.

ac0a0c4d-6b7e-405e-a968-7960719d9f42-0
00:08:47.640 --> 00:08:52.600
So you can have multiple batch
steps within the process phase

ac0a0c4d-6b7e-405e-a968-7960719d9f42-1
00:08:52.600 --> 00:08:55.880
and a batch step also has an
aggregator.

7299b69e-797a-428a-8ed9-8dfc2a1e6791-0
00:08:56.320 --> 00:08:58.840
Again, I won't talk much about
it in the PPT.

e1f59fe0-ca59-46a0-9462-c8c178efd079-0
00:08:58.840 --> 00:09:01.799
So we'll see a demo like how the
aggregator works, What is the

e1f59fe0-ca59-46a0-9462-c8c178efd079-1
00:09:01.799 --> 00:09:02.880
need of the aggregator?

7ae7b357-84c7-4b94-b88d-e0378a11c8e2-0
00:09:03.200 --> 00:09:06.229
OK, so it's, as the name
implies, it's going to aggregate

7ae7b357-84c7-4b94-b88d-e0378a11c8e2-1
00:09:06.229 --> 00:09:09.520
X amount of records and process
them again as a single record.

b2776ae8-4197-41e5-bcad-cff53c5b89d4-0
00:09:09.760 --> 00:09:11.720
So we'll talk more about it as
we move forward.

3335916c-7512-4cd5-b0d6-4d5eaa9787a1-0
00:09:12.440 --> 00:09:14.520
And finally you have the on
complete phase.

f2cbf56b-e65a-46b5-b382-55534418d51c-0
00:09:14.720 --> 00:09:18.440
Again the on complete phase, you
don't have to do anything.

5f7e4ab5-8e6d-496e-99bc-057665d4b4ea-0
00:09:18.640 --> 00:09:22.781
It only gives you a report which
contains the details about the

5f7e4ab5-8e6d-496e-99bc-057665d4b4ea-1
00:09:22.781 --> 00:09:26.793
batch job like how many records
were processed, how many were

5f7e4ab5-8e6d-496e-99bc-057665d4b4ea-2
00:09:26.793 --> 00:09:28.800
successful and how many failed.

bcdf5a94-d058-498f-a0f9-7709cac07d5e-0

00:09:29.080 --> 00:09:33.190
OK, so those details will be you
know provided automatically non

bcdf5a94-d058-498f-a0f9-7709cac07d5e-1
00:09:33.190 --> 00:09:33.760
complete.

21efc6ed-1531-4931-929d-05b74cba4ce0-0
00:09:33.760 --> 00:09:37.088
Even if you don't give anything
here it is going to be you know

21efc6ed-1531-4931-929d-05b74cba4ce0-1
00:09:37.088 --> 00:09:38.960
implicitly providing those
details.

47563a63-64e3-4f42-845c-f35a55ef92db-0
00:09:39.160 --> 00:09:42.184
So as a developer, you have to
focus more on the processing

47563a63-64e3-4f42-845c-f35a55ef92db-1
00:09:42.184 --> 00:09:45.057
phase because that is a place
where you add the business

47563a63-64e3-4f42-845c-f35a55ef92db-2
00:09:45.057 --> 00:09:45.359
logic.

b9040eb7-f0f2-4a60-bf8e-e670fab3a410-0
00:09:45.600 --> 00:09:45.920
OK.

0dc2a0ca-e01f-4603-a8c5-07ccc9b28bb9-0
00:09:46.480 --> 00:09:47.080
So let me see.

6539e093-d58b-4dc7-9923-52a294e7d067-0
00:09:47.080 --> 00:09:48.720
I mean, I hear a lot of pinks.

1cc11482-d5af-46ac-8e24-3639915dab40-0
00:09:48.720 --> 00:09:51.000
I guess people are joining in.

cc4b642c-3c8d-4fce-af8e-c1431d538627-0
00:09:51.080 --> 00:09:53.360
OK, OK.

22da3422-35f2-4c2f-9f3e-ebee1fb7e5c8-0
00:09:53.360 --> 00:09:54.720
So that's the three phases.

349e78d7-58c7-4309-93ab-39ed2b6ad377-0
00:09:55.120 --> 00:09:59.480
So now let me explain how batch
job works.

8bacc7e8-5a87-40a0-9212-c0c0419ca5ae-0
00:09:59.640 --> 00:09:59.960
OK.

4c3d7fd9-9394-402c-8f9e-ef2b65676834-0
00:10:00.280 --> 00:10:04.200
So within a batch job, So OK, I
think this slide is misleading.

f15a7175-80ca-49f1-9dc8-56eefd7d9f58-0
00:10:04.200 --> 00:10:05.680
So let me just modify.

f9ff2adc-628a-4d0d-ba50-078cdbb5b4ab-0
00:10:05.680 --> 00:10:09.853
I think I thought of modifying
this, but yeah, it should be

f9ff2adc-628a-4d0d-ba50-078cdbb5b4ab-1
00:10:09.853 --> 00:10:10.480
modified.

3baffd83-5519-47e2-981b-427307e43bb3-0
00:10:10.520 --> 00:10:14.160
But multiple things are there.

957161a8-4b3c-4937-8e34-21428fd2c184-0
00:10:15.240 --> 00:10:16.600
Just give me one second.

7d4fc817-84ee-40f8-b58d-fa10e0d42748-0
00:10:32.040 --> 00:10:34.760
Yeah, the PPT is broken.

a4735ef6-6705-49c6-9627-f96ed16fc4ae-0
00:10:35.240 --> 00:10:38.408
It's a bit difficult to explain
the concept, so just give me a

a4735ef6-6705-49c6-9627-f96ed16fc4ae-1
00:10:38.408 --> 00:10:38.760
minute.

bd4c4e37-b553-4067-b662-7ee87ef6c4b7-0
00:10:39.760 --> 00:10:40.440
OK?

64f9289a-3b81-4bb7-9a48-3027aa83fd08-0
00:10:40.440 --> 00:10:42.480
I had to do all this animations.

9547ab1f-f22e-454e-8ff4-85635ecdff46-0
00:10:43.680 --> 00:10:44.480
Yeah, that's great.

8657eeae-b58f-4352-b0b0-9cfbc007dc5f-0
00:10:46.160 --> 00:10:48.400
OK, so let's understand the
third slide.

d9d7807d-da0b-4bad-b22a-13e4ababa777-0
00:10:48.720 --> 00:10:53.340
So here in the batch job you
have several fields which can be

d9d7807d-da0b-4bad-b22a-13e4ababa777-1
00:10:53.340 --> 00:10:54.160
configured.

9810b12b-2007-4da2-bf08-9b130051eb15-0
00:10:54.520 --> 00:10:57.280
So the first one is the name of
the batch job.

366a345c-e049-46da-9ca2-e97506f4824f-0
00:10:57.280 --> 00:10:59.842
So whatever you see batch or
disco job so that is the name

366a345c-e049-46da-9ca2-e97506f4824f-1
00:10:59.842 --> 00:11:00.320
given here.

531beab4-49e0-460b-bbbe-a6146946657d-0
00:11:00.680 --> 00:11:03.080
So we have something called as
Max failed records.

c4b6be45-244a-4d9a-8704-fba8cff50501-0
00:11:03.080 --> 00:11:06.272
Again I will not talk about the
Max failed records now because

c4b6be45-244a-4d9a-8704-fba8cff50501-1
00:11:06.272 --> 00:11:07.640
it will not make any sense.

4700072f-a365-48a1-9a99-4d0616f20c54-0
00:11:07.800 --> 00:11:10.000
I'll do a demo on that.

eadb49da-b846-42e9-9dcd-bc53de8f26ab-0
00:11:10.560 --> 00:11:12.840
And then again there is
something called as.

25f842de-6d32-45ed-8700-7dd07828bd9e-0
00:11:13.720 --> 00:11:15.480
Let me just add a pointer.

fe82a58a-fddf-415c-a57e-d722cb27ccb4-0
00:11:16.440 --> 00:11:18.040
There's something called a
scheduling strategy.

1c77c14a-1fbc-4b0c-b309-4eaf4d13bece-0
00:11:18.040 --> 00:11:21.231
So you have two types of
scheduling strategies,

1c77c14a-1fbc-4b0c-b309-4eaf4d13bece-1
00:11:21.231 --> 00:11:22.960
sequential or round Robin.

3ca47ed5-f59a-4c91-88df-474994cc0e5f-0
00:11:22.960 --> 00:11:25.520
So again this I will be covering
as part of the demo.

b572dcdc-544f-44fa-9ead-2d82049a6d0e-0
00:11:25.960 --> 00:11:29.240
But now I want to talk about the
job instance ID.

d7565166-26ee-4818-be34-8adaa1efebc8-0
00:11:29.560 --> 00:11:31.760
So what is this job instance ID?

ee8c58e2-afba-4efd-979d-ce794400e109-0
00:11:31.760 --> 00:11:37.126
So this is nothing but a unique
ID which will be auto generated

ee8c58e2-afba-4efd-979d-ce794400e109-1
00:11:37.126 --> 00:11:38.720
for each batch job.

f976eb36-1e01-428e-8204-da01461e4297-0
00:11:39.440 --> 00:11:42.664
OK so the reason why this ID is
required is so you I mean you

f976eb36-1e01-428e-8204-da01461e4297-1
00:11:42.664 --> 00:11:45.940
can also hard code something or
you can dynamically generate a

f976eb36-1e01-428e-8204-da01461e4297-2
00:11:45.940 --> 00:11:46.200
UUID.

2709d653-aa00-49b2-ac49-737ec6a290a2-0
00:11:46.560 --> 00:11:50.076
So if you remember what happens
is when you first make a call to

2709d653-aa00-49b2-ac49-737ec6a290a2-1
00:11:50.076 --> 00:11:52.240
this flow the batch job gets
initiated.

320871aa-ad19-4906-9426-dd281331c100-0
00:11:52.400 --> 00:11:55.232
When the batch job gets
initiated the first phase is a

320871aa-ad19-4906-9426-dd281331c100-1
00:11:55.232 --> 00:11:56.160
load and dispatch.

52f5dfd0-9379-4817-8791-acc69f569449-0
00:11:56.160 --> 00:11:59.029
In the load and dispatch, if you
remember, it creates A

52f5dfd0-9379-4817-8791-acc69f569449-1
00:11:59.029 --> 00:12:02.104
persistent VMQ and stores all
the incoming thousand records

52f5dfd0-9379-4817-8791-acc69f569449-2
00:12:02.104 --> 00:12:04.000
and then the processing will
happen.

cfc92e5d-f708-4909-9011-e74fa2354913-0
00:12:04.360 --> 00:12:08.006
So let's assume you're
processing a million records and

cfc92e5d-f708-4909-9011-e74fa2354913-1
00:12:08.006 --> 00:12:11.719
it might take let's say 2 hours
to process those million

cfc92e5d-f708-4909-9011-e74fa2354913-2
00:12:11.719 --> 00:12:12.239
records.

68ccb76f-4047-4bdc-a6dd-ed22762985b3-0
00:12:12.680 --> 00:12:16.135
So within this two hours you
might make another call to do

68ccb76f-4047-4bdc-a6dd-ed22762985b3-1
00:12:16.135 --> 00:12:17.600
another batch processing.

d8514bbc-99ec-42f8-90b9-a13177182ee1-0
00:12:17.720 --> 00:12:20.360
So when or or OK instead of
another call.

a327e664-08f9-43f1-94c4-f9c91cead5d2-0
00:12:20.360 --> 00:12:24.393
Let's assume within the same
application, you might be having

a327e664-08f9-43f1-94c4-f9c91cead5d2-1
00:12:24.393 --> 00:12:26.800
another flow with another batch
job.

99ef85b1-7144-4d7d-a6dc-4679678b6f95-0
00:12:27.280 --> 00:12:30.737
So if this batch stop is also
executed, so there will be

99ef85b1-7144-4d7d-a6dc-4679678b6f95-1
00:12:30.737 --> 00:12:34.315
another VMQ which will be
created which will store all the

99ef85b1-7144-4d7d-a6dc-4679678b6f95-2
00:12:34.315 --> 00:12:36.560
values for the 2nd batch job,
right?

cd26ba6f-0064-4520-9cdf-1cfe668e31f2-0
00:12:36.880 --> 00:12:40.330
So let's say the processing is
happening fine but there is a

cd26ba6f-0064-4520-9cdf-1cfe668e31f2-1
00:12:40.330 --> 00:12:42.480
crash, the application got
restarted.

58a3b71f-f4de-4a0b-b3e0-06b62729d2d6-0
00:12:42.680 --> 00:12:45.978
When the application got
restarted, as I mentioned

58a3b71f-f4de-4a0b-b3e0-06b62729d2d6-1

00:12:45.978 --> 00:12:49.858
earlier, the batch job is going
to resume from where it was

58a3b71f-f4de-4a0b-b3e0-06b62729d2d6-2
00:12:49.858 --> 00:12:53.544
stopped because that is the
major use case of batch jobs

58a3b71f-f4de-4a0b-b3e0-06b62729d2d6-3
00:12:53.544 --> 00:12:54.320
reliability.

5f3286d0-b45c-4e16-a576-dd92b5c206d4-0
00:12:54.640 --> 00:12:58.096
But here there might be a
confusion because you have two

5f3286d0-b45c-4e16-a576-dd92b5c206d4-1
00:12:58.096 --> 00:13:00.280
persistent VMQS and two batch
jobs.

62587290-46a2-42ee-ab1e-cd2510d11e10-0
00:13:00.400 --> 00:13:04.570
So how does the Mule Runtime
engine associate the persistent

62587290-46a2-42ee-ab1e-cd2510d11e10-1
00:13:04.570 --> 00:13:06.280
VMQS with the batch jobs?

ec2ebf89-ba8d-4087-a58a-7322536dee82-0
00:13:06.920 --> 00:13:10.144
So that is the place where the
job instance ID comes into the

ec2ebf89-ba8d-4087-a58a-7322536dee82-1
00:13:10.144 --> 00:13:10.560
picture.

522d41fd-7b23-4b0a-b16a-d393ff555e91-0
00:13:10.840 --> 00:13:15.747
So each batch job has a unique
job instance ID which will be

522d41fd-7b23-4b0a-b16a-d393ff555e91-1
00:13:15.747 --> 00:13:18.080
mapped to the persistent VMQ.

8ae3fa09-ad98-4ba3-9dd1-0726053f134b-0
00:13:18.160 --> 00:13:22.986
So each queue will or each you
know VMQ will be mapped to 1

8ae3fa09-ad98-4ba3-9dd1-0726053f134b-1
00:13:22.986 --> 00:13:25.560
batch job using the instance ID.

c6e0ee8c-1968-4e1f-ba84-c6dfba6ed5c5-0
00:13:25.960 --> 00:13:29.606
OK, so that is how even if you
have multiple batch jobs within

c6e0ee8c-1968-4e1f-ba84-c6dfba6ed5c5-1

00:13:29.606 --> 00:13:33.080
the same application, you know
the data won't be corrupted.

9ab4631b-c4d4-4caf-b0da-3aedeb553eba-0
00:13:33.080 --> 00:13:36.789
So they know you know which VMQ
you know has to be used for

9ab4631b-c4d4-4caf-b0da-3aedeb553eba-1
00:13:36.789 --> 00:13:37.840
pulling the data.

583d3b7d-884f-434a-bbe5-c817f9030fb1-0
00:13:38.720 --> 00:13:41.360
OK, so that's about the job
instance ID.

2693a0ac-6c58-444b-9584-afeac9398730-0
00:13:41.760 --> 00:13:45.738
And then I want to talk about
the batch block size and the Max

2693a0ac-6c58-444b-9584-afeac9398730-1
00:13:45.738 --> 00:13:49.654
concurrency and I'll I also want
to discuss how the execution

2693a0ac-6c58-444b-9584-afeac9398730-2
00:13:49.654 --> 00:13:50.160
happens.

032e75ab-37c4-4b58-bb0e-5c97ff203891-0
00:13:50.560 --> 00:13:52.240
So let me come to this slide.

848a1cca-d645-4c70-a78a-0516e592d782-0
00:13:52.280 --> 00:13:54.680
So let's take the same example.

3c502dc6-0b7e-4318-a8b9-ada56c7c276f-0
00:13:55.080 --> 00:13:59.610
So we have a batch job, we have
two steps, batch step one and

3c502dc6-0b7e-4318-a8b9-ada56c7c276f-1
00:13:59.610 --> 00:14:00.560
batch Step 2.

10c12f90-46fa-4e7a-9f85-435bff5fd122-0
00:14:01.080 --> 00:14:05.280
And let's assume the input
payload sent thousand records.

1a4d0cc9-966a-4c4a-901f-c39ffc0d3ab9-0
00:14:05.280 --> 00:14:09.160
So all the 1000 records are
persisted in a VMQ.

2b3cf40a-59ad-4551-bc29-0952380dc27c-0
00:14:09.600 --> 00:14:11.480
So now how the processing
happens?

afe8a4e4-5eaa-4db4-bae9-26f8cb96fced-0

00:14:11.920 --> 00:14:14.760
OK so this persistent VMQ.

7762cb56-5d12-4e20-990b-c78a42ec10d1-0
00:14:14.760 --> 00:14:19.276
So if you use any point studio
this will be returned to a file

7762cb56-5d12-4e20-990b-c78a42ec10d1-1
00:14:19.276 --> 00:14:23.864
on the local machine and if you
use cloud hub it is going to be

7762cb56-5d12-4e20-990b-c78a42ec10d1-2
00:14:23.864 --> 00:14:25.799
persisted using Amazon SQS.

dbfe26ce-777e-4446-8e0a-11dc27a7337e-0
00:14:25.800 --> 00:14:29.390
I hope you remember this points
when I talked about VMQS maybe

dbfe26ce-777e-4446-8e0a-11dc27a7337e-1
00:14:29.390 --> 00:14:29.960
last week.

456fb1f2-5af9-4b4b-aeec-fc40d7ec9a60-0
00:14:30.040 --> 00:14:33.742
OK, so whenever the application
wants to fetch the data, yeah

456fb1f2-5af9-4b4b-aeec-fc40d7ec9a60-1
00:14:33.742 --> 00:14:34.160
thanks.

b4160145-3055-41f7-b8b4-793b3ac3c0a9-0
00:14:34.440 --> 00:14:38.257
So it has to connect with the
file or to the Amazon SQS to

b4160145-3055-41f7-b8b4-793b3ac3c0a9-1
00:14:38.257 --> 00:14:38.840
fetch it.

38528878-3d11-44a1-b3f4-36d36d90cee7-0
00:14:39.080 --> 00:14:43.249
So so you you have complete
control on how to fetch and how

38528878-3d11-44a1-b3f4-36d36d90cee7-1
00:14:43.249 --> 00:14:45.960
many records to be fetched to
process.

5a50bbf1-0f4b-41a7-951f-7ff61529cfdf-0
00:14:46.280 --> 00:14:49.840
So that is defined using the
batch block size.

86a1f316-5382-4fd5-b33b-0400694738c2-0
00:14:50.040 --> 00:14:52.280
So the default block size is
100.

a7bfbae2-ad9d-4539-968e-8824c2a85c9e-0

00:14:52.520 --> 00:14:56.411
So when I say 100, it's going to
load 100 records from the VMQ

a7bfbae2-ad9d-4539-968e-8824c2a85c9e-1
00:14:56.411 --> 00:14:59.870
into the memory, into the
application's memory and then

a7bfbae2-ad9d-4539-968e-8824c2a85c9e-2
00:14:59.870 --> 00:15:02.280
it's going to process the 100
records.

2464ff00-e5bb-4e59-b793-3e6fc5818244-0
00:15:02.560 --> 00:15:07.200
So to make things simpler, I'll
just OK, where is this?

84893aa4-d678-400f-b7a6-6b781f6fe5c3-0
00:15:07.200 --> 00:15:11.437
Yeah, so to make things simpler,
I just want to use batch block

84893aa4-d678-400f-b7a6-6b781f6fe5c3-1
00:15:11.437 --> 00:15:12.960
sizes two I customized.

75bf4abe-f691-410d-aa15-78b9c59f75de-0
00:15:13.000 --> 00:15:16.616
You can customize this value and
I just want to use the block

75bf4abe-f691-410d-aa15-78b9c59f75de-1
00:15:16.616 --> 00:15:17.200
sizes two.

8f6d5af5-7dbc-4caa-b889-859c61849cc5-0
00:15:17.920 --> 00:15:18.240
OK.

e96980b2-25e7-41e6-8091-1f214dc7108a-0
00:15:18.520 --> 00:15:21.846
And moreover, when you deploy
the application based on the

e96980b2-25e7-41e6-8091-1f214dc7108a-1
00:15:21.846 --> 00:15:25.228
number of CPUs, I mean based on
the number of CPU cores and

e96980b2-25e7-41e6-8091-1f214dc7108a-2
00:15:25.228 --> 00:15:28.780
memory, it's going to allocate,
you know some threads for your

e96980b2-25e7-41e6-8091-1f214dc7108a-3
00:15:28.780 --> 00:15:31.994
application and this is
automatic process I think I also

e96980b2-25e7-41e6-8091-1f214dc7108a-4
00:15:31.994 --> 00:15:32.840
discussed this.

6f3bfde2-36a7-4582-9fcc-a5ca7a08f61a-0
00:15:33.080 --> 00:15:36.393
So this is again automated
process where the application

6f3bfde2-36a7-4582-9fcc-a5ca7a08f61a-1
00:15:36.393 --> 00:15:39.822
based on your computer's CPU
course it's going to allocate

6f3bfde2-36a7-4582-9fcc-a5ca7a08f61a-2
00:15:39.822 --> 00:15:40.520
the threads.

710c67b9-de0a-4c37-bddb-ecda7053b98f-0
00:15:40.520 --> 00:15:43.676
OK so this in the thread pool
you'll be having all the threads

710c67b9-de0a-4c37-bddb-ecda7053b98f-1
00:15:43.676 --> 00:15:45.680
which can be used for this
application.

67facf0b-5c0e-42da-9d6f-ed04d51ea9f5-0
00:15:46.280 --> 00:15:50.604
So since the batch block sizes
two, what this batch dog is

67facf0b-5c0e-42da-9d6f-ed04d51ea9f5-1
00:15:50.604 --> 00:15:54.635
going to do is it is going to
fetch 2 records from the

67facf0b-5c0e-42da-9d6f-ed04d51ea9f5-2
00:15:54.635 --> 00:15:57.639
persistent VMQ and store them in
memory.

0c99f6b6-5bee-4cbb-83a6-412e40fdb4d0-0
00:15:58.320 --> 00:16:01.280
So this in memory is where the
application is running.

2635861a-f8a8-4c31-90c4-4d65935021fa-0
00:16:01.280 --> 00:16:05.729
So this is the application's RAM
where the records have been

2635861a-f8a8-4c31-90c4-4d65935021fa-1
00:16:05.729 --> 00:16:06.240
stored.

e36031f2-e193-4638-87d0-610f6d79d0b8-0
00:16:06.640 --> 00:16:10.908
So once the records are stored
you can also define Max

e36031f2-e193-4638-87d0-610f6d79d0b8-1
00:16:10.908 --> 00:16:11.840
concurrency.

fbc5a98c-c2c4-42e5-9b56-02bb4621590e-0
00:16:12.000 --> 00:16:14.680
So Max concurrency is the.

8bbb225d-236a-4bd3-b8bd-5abf1961b4c3-0
00:16:15.160 --> 00:16:19.640
I know how many records you want
to process in parallel and the

8bbb225d-236a-4bd3-b8bd-5abf1961b4c3-1
00:16:19.640 --> 00:16:21.600
value of Max concurrency is.

c549dfce-118e-46e2-885e-df95445d8c7b-0
00:16:21.800 --> 00:16:25.760
It is always equal to two times
the number of CPU cores.

714b6379-a397-435d-a048-86a874ab4653-0
00:16:25.880 --> 00:16:30.721
So if you have 8 CPU cores on
your machine, it can use 16

714b6379-a397-435d-a048-86a874ab4653-1
00:16:30.721 --> 00:16:35.480
threads so it can process 16 you
know records at a time.

33c21590-c757-45e0-9af5-3c7b057634bd-0
00:16:35.960 --> 00:16:39.796
So if you have 100 records in
memory so it can process 16

33c21590-c757-45e0-9af5-3c7b057634bd-1
00:16:39.796 --> 00:16:41.120
records in parallel.

ed316a28-1771-405f-a423-d2c0964e6add-0
00:16:41.280 --> 00:16:43.760
So that is how the Max
concurrency is decided.

0be5dd59-e584-4070-839c-962766f81f3a-0
00:16:44.360 --> 00:16:48.524
OK, so in our example, let's
assume the Max concurrency is

0be5dd59-e584-4070-839c-962766f81f3a-1
00:16:48.524 --> 00:16:49.160
also two.

0b2fe94c-9c8d-48c3-be6d-b75bc13abb4c-0
00:16:49.280 --> 00:16:51.840
OK, I didn't define two, but
let's assume it is 2.

4ca644e1-eadc-464d-b451-3272279cbe2b-0
00:16:52.120 --> 00:16:56.260
So if the Max concurrency is 2,
it is going to use two threads

4ca644e1-eadc-464d-b451-3272279cbe2b-1
00:16:56.260 --> 00:16:57.640
from the thread pool.

c9811280-904d-4566-ae99-29226143cd80-0
00:16:58.000 --> 00:17:00.360
So the two threads go to the in

memory.

a00f8b81-4692-422a-a983-7854d3560599-0
00:17:00.520 --> 00:17:04.560
It's going to, you know, fetch
both the records which are there

a00f8b81-4692-422a-a983-7854d3560599-1
00:17:04.560 --> 00:17:08.600
in memory and those records will
be sent to the batch step one.

ece4133b-59ff-4ce7-a416-390a325e16d6-0
00:17:09.040 --> 00:17:12.855
So in the batch step one, both
the records will be executed in

ece4133b-59ff-4ce7-a416-390a325e16d6-1
00:17:12.855 --> 00:17:13.400
parallel.

51443308-d58e-4feb-a427-5c54876c7b40-0
00:17:13.880 --> 00:17:18.296
If the Max concurrency is 1616
records will be processed in

51443308-d58e-4feb-a427-5c54876c7b40-1
00:17:18.296 --> 00:17:19.400
parallel right?

9d333cde-3039-4ee7-8a9a-6c62034ecd68-0
00:17:19.400 --> 00:17:22.726
So it totally depends upon how
many CPU codes you have and how

9d333cde-3039-4ee7-8a9a-6c62034ecd68-1
00:17:22.726 --> 00:17:25.947
many you know, threads you want
to use, how many records you

9d333cde-3039-4ee7-8a9a-6c62034ecd68-2
00:17:25.947 --> 00:17:27.320
want to parallely process.

1c2e67c9-c89f-4fe1-8355-1d23a42b03c2-0
00:17:27.680 --> 00:17:29.680
OK, so I'm just taking two as an
example.

d20a5739-9562-45e2-95cd-47559212c6c4-0
00:17:29.920 --> 00:17:34.099
So here let's assume both the
records are being processed and

d20a5739-9562-45e2-95cd-47559212c6c4-1
00:17:34.099 --> 00:17:37.200
the second record completed
processing first.

caa05737-7614-4b4a-85ef-fa4f449bc00f-0
00:17:37.200 --> 00:17:40.134
I mean I mean completed the
processing even before the 1st

caa05737-7614-4b4a-85ef-fa4f449bc00f-1

00:17:40.134 --> 00:17:40.880
record is done.

55ef2f07-c757-4507-9797-fd9f7369ec9f-0
00:17:41.160 --> 00:17:44.894
So when the second record
processing is over, it is again

55ef2f07-c757-4507-9797-fd9f7369ec9f-1
00:17:44.894 --> 00:17:48.886
going to be pushed back to the
original VMQ or the persistent

55ef2f07-c757-4507-9797-fd9f7369ec9f-2
00:17:48.886 --> 00:17:52.878
VMQ and the first record will
still be processing in step one

55ef2f07-c757-4507-9797-fd9f7369ec9f-3
00:17:52.878 --> 00:17:55.840
and the same process repeats
again and again.

79f4b393-1607-4834-bb9d-86eff5ecf479-0
00:17:56.000 --> 00:17:59.795
But the next time the second
record it will go to the second

79f4b393-1607-4834-bb9d-86eff5ecf479-1
00:17:59.795 --> 00:18:00.480
batch step.

e4e56a8d-6b45-42d1-b718-e62cc840e693-0
00:18:01.000 --> 00:18:04.612
So how does the batch job knows
that the second record has to go

e4e56a8d-6b45-42d1-b718-e62cc840e693-1
00:18:04.612 --> 00:18:07.280
to the second batch step and not
the first one?

128dd426-88c3-4373-85ac-d64fff83925f-0
00:18:07.920 --> 00:18:10.200
So that is stored in the
metadata.

32bf1d01-14fc-47ef-b46a-d6e936fd4428-0
00:18:10.200 --> 00:18:13.560
If you see each record can also
hold some metadata.

f2ac424f-abd0-4e9f-9342-f27901c239d3-0
00:18:13.840 --> 00:18:17.446
That metadata contains the
information on what step this

f2ac424f-abd0-4e9f-9342-f27901c239d3-1
00:18:17.446 --> 00:18:20.040
record has already completed
processing.

66e1bee3-3019-4690-bd57-80a52bfc7b3f-0
00:18:20.280 --> 00:18:23.417
So based on the metadata, the

batch stop is automatically

66e1bee3-3019-4690-bd57-80a52bfc7b3f-1
00:18:23.417 --> 00:18:26.662
going to, you know, process the
second record in the second

66e1bee3-3019-4690-bd57-80a52bfc7b3f-2
00:18:26.662 --> 00:18:29.799
batch step because it completed
processing the first one.

196e15e7-8784-4e4f-ac66-1ac15efba56a-0
00:18:30.320 --> 00:18:33.477
So here again since the records
are being processed parallely,

196e15e7-8784-4e4f-ac66-1ac15efba56a-1
00:18:33.477 --> 00:18:36.083
it doesn't matter even if the
first record is still

196e15e7-8784-4e4f-ac66-1ac15efba56a-2
00:18:36.083 --> 00:18:39.040
processing, the other records
can complete the processing.

8746f03c-a64b-4af8-ab5c-dde3fa6ce053-0
00:18:39.560 --> 00:18:43.495
OK, so that is how the records
will be pulled from VMQ, stored

8746f03c-a64b-4af8-ab5c-dde3fa6ce053-1
00:18:43.495 --> 00:18:47.118
in memory and the threads are
going to you know use those

8746f03c-a64b-4af8-ab5c-dde3fa6ce053-2
00:18:47.118 --> 00:18:48.680
records and process them.

79fd8697-b492-4c51-ac0c-a2bba8fd872d-0
00:18:49.120 --> 00:18:52.260
And once the second record is
completing processing in batch

79fd8697-b492-4c51-ac0c-a2bba8fd872d-1
00:18:52.260 --> 00:18:54.680
step two, we don't have any
other batch steps.

9b4e67a9-85a6-4902-b9a8-7601e303761f-0
00:18:54.840 --> 00:18:58.466
So what happens is it is going
to simply delete the record from

9b4e67a9-85a6-4902-b9a8-7601e303761f-1
00:18:58.466 --> 00:18:58.920
the VMQ.

dd1bc26e-0611-4c36-958a-56eff6f7197f-0
00:18:59.400 --> 00:19:03.591
So once all the batch steps are
completed, the records are

dd1bc26e-0611-4c36-958a-56eff6f7197f-1
00:19:03.591 --> 00:19:04.160
deleted.

56676910-372f-47c2-b728-d2a00d8cb111-0
00:19:04.520 --> 00:19:08.480
OK, so that's how batch
processing works.

81f1576f-4180-4c21-8f64-d5d02a31dcc6-0
00:19:09.120 --> 00:19:09.920
Any questions?

d7e63fb3-6059-4655-8271-1f538632b2fd-0
00:19:15.730 --> 00:19:17.090
So one quick question.

be851f3d-5439-4dd9-94d3-27935aa5f3f8-0
00:19:17.090 --> 00:19:19.130
So that so you said the batch
steps, right?

081450a4-25a8-42c5-b4f9-5859c99fadbc-0
00:19:19.370 --> 00:19:21.653
We have multiple, say new
example we have in the given

081450a4-25a8-42c5-b4f9-5859c99fadbc-1
00:19:21.653 --> 00:19:23.730
example, there are only two
batch steps, correct?

33135401-e053-4979-b379-41f3c51ccbb9-0
00:19:24.170 --> 00:19:24.810
Yes, yes.

37f8c19d-ef75-4cee-b5c8-fad84d634738-0
00:19:25.330 --> 00:19:28.463
So, so you have the first
record, say you have in memory

37f8c19d-ef75-4cee-b5c8-fad84d634738-1
00:19:28.463 --> 00:19:31.872
you have two records and 1st
record is still processing batch

37f8c19d-ef75-4cee-b5c8-fad84d634738-2
00:19:31.872 --> 00:19:35.280
Step 2 and second record is a
completed batch step batch Step

37f8c19d-ef75-4cee-b5c8-fad84d634738-3
00:19:35.280 --> 00:19:36.600
2 ahead of first record.

cc3566b0-7a65-4569-bac4-ea3136140f8f-0
00:19:36.600 --> 00:19:39.342
So it will again go back to the
same because you don't have any

cc3566b0-7a65-4569-bac4-ea3136140f8f-1
00:19:39.342 --> 00:19:40.200
further steps right?

0f9db9ef-39cf-427e-927b-c3d702600e78-0

00:19:40.200 --> 00:19:42.520
So I just read it huh?

0d886d93-1987-4059-a7c0-33e6a8aaf54f-0
00:19:42.560 --> 00:19:45.681
I mean no no if the if the
record completes all the batch

0d886d93-1987-4059-a7c0-33e6a8aaf54f-1
00:19:45.681 --> 00:19:49.072
steps which are available in the
job, it's going to be deleted

0d886d93-1987-4059-a7c0-33e6a8aaf54f-2
00:19:49.072 --> 00:19:49.880
from the queue.

1c13b35b-1655-46fb-aa97-4c9b52b91a5a-0
00:19:52.680 --> 00:19:56.495
So OK I'm just asking that might
be yeah because what are the job

1c13b35b-1655-46fb-aa97-4c9b52b91a5a-1
00:19:56.495 --> 00:19:59.560
is done that writing will DB are
OK, OK recorded so.

9c4ce7cb-21a2-4e48-9b0e-92579348cd6e-0
00:20:02.000 --> 00:20:04.753
And what I mean to say, you have
a sequential writing to a file

9c4ce7cb-21a2-4e48-9b0e-92579348cd6e-1
00:20:04.753 --> 00:20:07.248
or appending to a, suppose
you're reading a file from one

9c4ce7cb-21a2-4e48-9b0e-92579348cd6e-2
00:20:07.248 --> 00:20:09.356
data and doing some
transformation, but the same

9c4ce7cb-21a2-4e48-9b0e-92579348cd6e-3
00:20:09.356 --> 00:20:11.119
level you have to sequentially
upgrades.

85eb7da4-e567-417c-ab72-2f25aae8470c-0
00:20:11.120 --> 00:20:13.880
And yeah, I'm just asking like
it would be in order, right.

78814831-b129-420a-a814-a0c499c53837-0
00:20:13.880 --> 00:20:17.187
So if you one, if you do that
concurrence, so depends

78814831-b129-420a-a814-a0c499c53837-1
00:20:17.187 --> 00:20:19.760
asynchronously, it won't be in
any order.

4c0b9805-6f70-4d4f-8e17-95f8421ef8a8-0
00:20:20.320 --> 00:20:21.840
OK, yeah, yeah.

2938876f-b289-4782-8bd0-fd70c5c72a72-0
00:20:21.840 --> 00:20:24.676
So if you want it in order, we
need to go with Farid's scope

2938876f-b289-4782-8bd0-fd70c5c72a72-1
00:20:24.676 --> 00:20:27.280
where it is going to process the
required sequentially.

1c48c74c-3fc0-4bd3-907f-6840fa97dcdb-0
00:20:28.480 --> 00:20:31.227
So here we cannot achieve that
even though it is like, yeah,

1c48c74c-3fc0-4bd3-907f-6840fa97dcdb-1
00:20:31.227 --> 00:20:33.120
right, let's see, you can't
achieve that.

c0e18824-6080-477b-b37c-ca499cfefb8f-0
00:20:33.200 --> 00:20:35.280
It's it's purely parallel,
asynchronous.

abe24e55-2283-46fb-904e-31f19d5848cb-0
00:20:35.280 --> 00:20:37.160
So so that's the reason I said
right.

0ae29a00-aa3c-4460-9bb2-e83d01b37b20-0
00:20:37.160 --> 00:20:39.541
So even if the first record is
not processed, the remaining

0ae29a00-aa3c-4460-9bb2-e83d01b37b20-1
00:20:39.541 --> 00:20:41.962
thousand records can complete
the processing and get deleted

0ae29a00-aa3c-4460-9bb2-e83d01b37b20-2
00:20:41.962 --> 00:20:42.280
as well.

22328f80-04fb-4a3f-aafc-c282c07c714d-0
00:20:42.560 --> 00:20:43.640
So it happens in parallel.

7c9a94b3-35dd-4105-b3ba-ea21edf50c83-0
00:20:43.640 --> 00:20:47.040
OK, OK, OK.

589d1807-d62e-43ce-a2b1-0c07e107404f-0
00:20:49.360 --> 00:20:54.442
So this Max concurrency, is it
automatically being fetched by

589d1807-d62e-43ce-a2b1-0c07e107404f-1
00:20:54.442 --> 00:20:56.000
the instance of or?

1248c82f-97c3-4da1-9775-56ffbbfe0b84-0
00:20:56.000 --> 00:21:00.671
I mean by by the application, I

mean you can define them

1248c82f-97c3-4da1-9775-56ffbbfe0b84-1
00:21:00.671 --> 00:21:05.425
manually, but if you don't
define anything, it is 2 times

1248c82f-97c3-4da1-9775-56ffbbfe0b84-2
00:21:05.425 --> 00:21:09.359
the number of CPU cores by
default, by default.

8bda66a2-b338-4ec6-bb1b-ce3fcce6a1b9-0
00:21:09.360 --> 00:21:12.636
So OK, if I if you see here, I
think in my machine I have

8bda66a2-b338-4ec6-bb1b-ce3fcce6a1b9-1
00:21:12.636 --> 00:21:16.308
around 4 cores, so it's going to
use Max concurrency as eight in

8bda66a2-b338-4ec6-bb1b-ce3fcce6a1b9-2
00:21:16.308 --> 00:21:16.760
my case.

0249ac40-bfd4-495f-b134-203ff535d825-0
00:21:16.800 --> 00:21:17.840
So that's how it works.

1611beb5-5f86-4bb2-ad58-f6e90d9f5434-0
00:21:18.840 --> 00:21:21.560
So yeah, I mean it's very slow.

bc5b5b76-9dd3-43e6-9983-b043714cb0b8-0
00:21:22.800 --> 00:21:27.271
But yeah, so if there's four
cores, it's going to use, you

bc5b5b76-9dd3-43e6-9983-b043714cb0b8-1
00:21:27.271 --> 00:21:30.000
know, eight as the Max
concurrency.

ff51d08f-0adc-41f5-9f30-097f1093091a-0
00:21:30.760 --> 00:21:32.080
So that's the default one.

cfdf58e9-fc92-411e-9aa9-c192bb63711b-0
00:21:32.280 --> 00:21:34.440
We can even have more values for
that.

52939969-09e6-408f-891a-db31079ccdd7-0
00:21:34.520 --> 00:21:38.427
You can have more values, but
the performance may be degraded

52939969-09e6-408f-891a-db31079ccdd7-1
00:21:38.427 --> 00:21:40.760
or it might improve the
performance.

f4ab3790-1a3e-4f74-a15a-e14b72a1b7b0-0
00:21:40.760 --> 00:21:44.424

So you have to do a lot of load
testing and a lot of testing has

f4ab3790-1a3e-4f74-a15a-e14b72a1b7b0-1
00:21:44.424 --> 00:21:47.525
to be done before you go to
production because Mulesop

f4ab3790-1a3e-4f74-a15a-e14b72a1b7b0-2
00:21:47.525 --> 00:21:50.908
doesn't recommend to use more
than the available, you know,

f4ab3790-1a3e-4f74-a15a-e14b72a1b7b0-3
00:21:50.908 --> 00:21:51.359
threads.

7c5aa8af-c3ee-4694-8fe0-2f2cb8989063-0
00:21:51.360 --> 00:21:54.266
So you have to do a lot of
testing and if it is working

7c5aa8af-c3ee-4694-8fe0-2f2cb8989063-1
00:21:54.266 --> 00:21:55.720
fine, yes, you can use that.

fba80d4b-dafc-42e2-bb1a-0779a8e201f7-0
00:21:57.200 --> 00:21:59.800
OK, OK.

51ee63ac-7011-4d56-93be-15d80b1b9e91-0
00:22:00.360 --> 00:22:03.040
And yeah, so that's about the
steps.

f419339d-3cb1-4864-9a2b-c5d2c191d81c-0
00:22:03.040 --> 00:22:04.920
So I think I just covered the
intro part.

a942d4f3-cb76-45b8-a816-e1a3cb3fb99b-0
00:22:04.920 --> 00:22:07.280
So let's try to understand this
using a demo.

59b956ac-5ab5-4e99-9ef2-ad180a172694-0
00:22:07.960 --> 00:22:11.120
So and again the intro, I think
I just completed the first step.

15b346a9-b3b5-4c45-9408-056004db0aaf-0
00:22:11.400 --> 00:22:14.120
OK, so let's go and do the demo
now.

22ff0c82-d1f7-4faa-b6bd-1bbb4c41c78e-0
00:22:14.600 --> 00:22:18.840
So if I go to my postman, I mean
sorry so I have a simple

22ff0c82-d1f7-4faa-b6bd-1bbb4c41c78e-1
00:22:18.840 --> 00:22:20.960
application already deployed.

439f5d36-b0ff-445f-94db-e3bb0209b1b4-0

00:22:21.360 --> 00:22:24.720
It is listening on slash batch.

41c95d4d-9fa5-4846-abed-6b2ce062d58f-0
00:22:24.920 --> 00:22:28.357
So ideally speaking we do not
use Http://listener for batch

41c95d4d-9fa5-4846-abed-6b2ce062d58f-1
00:22:28.357 --> 00:22:31.967
processing because batch is a
synchronous process whereas HTTP

41c95d4d-9fa5-4846-abed-6b2ce062d58f-2
00:22:31.967 --> 00:22:33.399
is synchronous connector.

89943630-e44f-4695-8cd3-c7ccae6012aa-0
00:22:33.640 --> 00:22:38.044
You ideally use scheduler or on
table row connectors where it

89943630-e44f-4695-8cd3-c7ccae6012aa-1
00:22:38.044 --> 00:22:42.520
can fetch the data or you know
execute this flow periodically.

bea89091-8605-407f-9dd8-4c1fca13e6bb-0
00:22:42.600 --> 00:22:46.053
OK, so just for demo purpose I'm
using Http://listener so that I

bea89091-8605-407f-9dd8-4c1fca13e6bb-1
00:22:46.053 --> 00:22:49.400
can control when to make a call
and when to do the processing.

7c455064-f735-4ef2-b8ea-c64cbdf93e74-0
00:22:49.880 --> 00:22:55.697
So instead of this logger, what
I want to do here is I'll just

7c455064-f735-4ef2-b8ea-c64cbdf93e74-1
00:22:55.697 --> 00:22:57.360
delete the logger.

be0d0aaa-89b9-450b-8301-a36b40e602a7-0
00:22:57.360 --> 00:22:58.120
I don't need it.

75d1b02f-fa6d-44ad-9d42-5cc4fd4f7ed3-0
00:22:58.720 --> 00:23:01.600
I go to core, I'll add a batch
job.

fecb06ac-d825-490b-80ab-febacc282123-0
00:23:02.240 --> 00:23:06.373
So when you add a batch job, by
default it has one batch step

fecb06ac-d825-490b-80ab-febacc282123-1
00:23:06.373 --> 00:23:10.240
which is in the process record
phase and one on complete.

ff0b5afe-18f0-4105-a056-ec0d5daab3e9-0
00:23:10.240 --> 00:23:16.120
OK, you can add multiple things
but first let me add a logger.

ff9d2d57-1833-4d4a-9df2-8d1b51d6c762-0
00:23:16.800 --> 00:23:18.640
So in the processor I'm adding a
logger.

f3bee959-6ad6-44db-9ff5-afe00aa4b2cc-0
00:23:18.640 --> 00:23:22.240
I just want to log the payload
for the first demo.

39a0328d-574b-451b-b069-93bd090297a8-0
00:23:22.880 --> 00:23:24.520
So let me see what demo I want
to do.

28d031e8-f1a3-4a05-9bda-925fb0a819ef-0
00:23:26.080 --> 00:23:30.190
So first I want to show you the
application logs using a simple

28d031e8-f1a3-4a05-9bda-925fb0a819ef-1
00:23:30.190 --> 00:23:30.640
logger.

0cede598-59e7-437c-bd6e-05a17f56376e-0
00:23:30.840 --> 00:23:34.160
OK, so I'll just use a simple
logger payload.

574da31c-7366-4e52-807f-2739e478d002-0
00:23:34.840 --> 00:23:36.080
Let me save this.

c6e67ffa-757c-40e0-acb8-436c44a8c260-0
00:23:37.200 --> 00:23:38.680
So let's add a break point.

b5805ee5-ef4b-4541-a663-83abd2f134be-0
00:23:40.680 --> 00:23:43.611
And yeah I mean if it if you
click on the batch job it's

b5805ee5-ef4b-4541-a663-83abd2f134be-1
00:23:43.611 --> 00:23:46.594
going to show you all the
options which I just showed you

b5805ee5-ef4b-4541-a663-83abd2f134be-2
00:23:46.594 --> 00:23:47.160
in the PPT.

dfa26a54-0bf9-4c49-9118-4778417108ae-0
00:23:47.360 --> 00:23:50.800
So we will be covering in a few
of them as we move forward.

31266b2f-58a5-4d61-bc2d-73064366a349-0
00:23:51.000 --> 00:23:54.240
OK, so let's go to the debug
mode.

694b500a-ebe4-462a-b5f5-9b9dce20f165-0
00:23:58.120 --> 00:24:00.909
Yeah, I'm not sure if I can
complete all this topics in the

694b500a-ebe4-462a-b5f5-9b9dce20f165-1
00:24:00.909 --> 00:24:01.560
next one hour.

e4a09bfe-02c9-43a3-9a8f-21260565fad9-0
00:24:02.080 --> 00:24:02.520
Let's see.

1163dd4b-8255-4391-bf73-f8b04fcf1905-0
00:24:06.040 --> 00:24:07.520
Yeah, it is deployed.

83770c07-6230-4d68-ba63-3b5808c20c68-0
00:24:07.800 --> 00:24:10.640
So now first let's check the
logs.

0645ae68-9e8e-4077-b5c6-ffc39a07630e-0
00:24:10.640 --> 00:24:14.400
OK, so when I make a call, so
let's go to Postman.

d343ffa7-c94b-486f-8186-de63649c5099-0
00:24:16.360 --> 00:24:19.842
So I have this, instead of 10
records, I'll just work with

d343ffa7-c94b-486f-8186-de63649c5099-1
00:24:19.842 --> 00:24:21.200
five records initially.

ea36611f-4b47-48a4-b377-14141105bb7f-0
00:24:21.840 --> 00:24:24.440
So let's say I have this file
record.

41dbc5e3-b15b-406f-bc0c-aaf0c786d63c-0
00:24:24.440 --> 00:24:27.800
So again batch job only accepts
area of records.

c0c7240c-88aa-424f-936f-0799eb629ada-0
00:24:27.880 --> 00:24:31.913
OK, you can't give it an object
so I'm just sending this area of

c0c7240c-88aa-424f-936f-0799eb629ada-1
00:24:31.913 --> 00:24:32.720
file records.

8149a672-8cdf-4d98-85ce-92f5f2de1625-0
00:24:33.080 --> 00:24:38.920
So let me just make a call so
the processing will stop here.

084a7e67-fef3-46bd-8686-11b0e627fd91-0
00:24:39.000 --> 00:24:42.097
But before I show you what is
happening in the batch tab, I

084a7e67-fef3-46bd-8686-11b0e627fd91-1
00:24:42.097 --> 00:24:43.440
want to show you the logs.

644129c4-4ed7-408c-8aaf-52a6cfdd2168-0
00:24:44.160 --> 00:24:49.560
So in the logs if you see it's
going to say start date.

74cd1806-ecbd-475e-bc96-e73f4cbf65a9-0
00:24:49.680 --> 00:24:50.800
OK, I think it's too small.

51bb54a0-cf50-472c-b35f-6a19cbf28b63-0
00:24:50.800 --> 00:24:52.160
Let me zoom this.

cec3d522-dcc4-41b5-bbed-7ea32f43b96b-0
00:24:55.600 --> 00:24:56.600
Yes, that should be fine.

a6b383d5-45aa-45f8-9e3f-718e8bf573f2-0
00:24:59.080 --> 00:25:01.760
Is it visible guys or do I need
to increase the font more?

a140d5d3-4e0a-475d-933b-83da62ec4a5f-0
00:25:02.280 --> 00:25:03.720
No, it's pretty much visible.

46dae2c8-11e5-4aec-907b-19621f82417f-0
00:25:04.000 --> 00:25:04.280
Visible.

09d744cb-fea2-49f4-a72b-22db55e9b8c0-0
00:25:04.440 --> 00:25:05.520
OK, Yeah.

27388b66-ddb4-4a69-b185-2768c3c4781b-0
00:25:05.680 --> 00:25:08.892
So the first thing is it's
saying that it is creating an

27388b66-ddb4-4a69-b185-2768c3c4781b-1
00:25:08.892 --> 00:25:09.400
instance.

5736e77f-708d-41d6-82a4-21450cf3b407-0
00:25:09.680 --> 00:25:12.652
So this is nothing but the job
instance ID, which is

5736e77f-708d-41d6-82a4-21450cf3b407-1
00:25:12.652 --> 00:25:15.120
automatically, you know, getting
generated.

2ee05fd5-c22a-4d17-8d92-c32398a59ec2-0
00:25:15.280 --> 00:25:15.440
OK.

31fa0dac-69e5-48b9-ad8a-af565caaa396-0
00:25:15.440 --> 00:25:18.560
Even if you don't define

anything, it's going to generate

31fa0dac-69e5-48b9-ad8a-af565caaa396-1
00:25:18.560 --> 00:25:20.120
a unique job instance ID, OK.

b4e804eb-d73d-4071-b397-fb6c876b864d-0
00:25:20.400 --> 00:25:23.157
And then it's saying that
starting loading phase, the

b4e804eb-d73d-4071-b397-fb6c876b864d-1
00:25:23.157 --> 00:25:26.118
first phase is the load and
dispatch phase, which happens

b4e804eb-d73d-4071-b397-fb6c876b864d-2
00:25:26.118 --> 00:25:26.680
implicitly.

abf1ea91-9471-404a-9449-ed271b91a6a2-0
00:25:26.680 --> 00:25:28.640
So this is happening
automatically.

859a4fc7-a0e9-4948-bc0e-72374c7eeb80-0
00:25:29.000 --> 00:25:29.360
OK.

644ca5e5-5bbe-47b7-9efb-87590db99c8e-0
00:25:29.640 --> 00:25:32.916
And it is saying that it is
finished the loading phase and

644ca5e5-5bbe-47b7-9efb-87590db99c8e-1
00:25:32.916 --> 00:25:36.193
if you see the end of the log,
it is saying 5 records were

644ca5e5-5bbe-47b7-9efb-87590db99c8e-2
00:25:36.193 --> 00:25:38.360
loaded because we only sent 5
records.

afec7e80-fc97-4593-af78-157dacf2035b-0
00:25:38.560 --> 00:25:41.840
So where was or where are the
records loaded?

ee8f5034-f114-41d1-adc2-49763502ae3b-0
00:25:42.080 --> 00:25:44.978
If you remember these are
persisted to a VM queue, so

ee8f5034-f114-41d1-adc2-49763502ae3b-1
00:25:44.978 --> 00:25:46.160
where is the VM queue?

76b58dcf-9647-4199-a868-4c036a5caea2-0
00:25:46.160 --> 00:25:50.979
So in our local machine since it
is any point studio you have to

76b58dcf-9647-4199-a868-4c036a5caea2-1
00:25:50.979 --> 00:25:55.280

go to, you have to go to any
point studio plug insurance.

bdbc5e8e-92be-4c2c-b0a2-caff90fcf9cd-0
00:25:55.960 --> 00:25:58.840
So it's going to use a file for
persisting the records.

abb32df1-8e7c-4d5c-a4e5-b461aff35753-0
00:25:59.320 --> 00:26:03.520
So you have to go to where is
this tooling server?

c58cb98d-60fa-46cc-8864-5bc482224cd2-0
00:26:03.520 --> 00:26:11.080
Yeah, I think it's tooling
server mule dot mule.

609f475e-a5ad-4d21-9f57-16a6c1c68f9c-0
00:26:14.160 --> 00:26:16.901
Yeah, I have this Merck batch
job which is the name of my

609f475e-a5ad-4d21-9f57-16a6c1c68f9c-1
00:26:16.901 --> 00:26:17.280
project.

99b1bb67-6a93-4e87-bced-8a361b900428-0
00:26:17.920 --> 00:26:21.698
Here you should be seeing
queues, queue, store and a queue

99b1bb67-6a93-4e87-bced-8a361b900428-1
00:26:21.698 --> 00:26:25.605
was created 6:25 PM just one
minute ago when I initiated the

99b1bb67-6a93-4e87-bced-8a361b900428-2
00:26:25.605 --> 00:26:29.575
call and we should be seeing I
think 3 queues will be created

99b1bb67-6a93-4e87-bced-8a361b900428-3
00:26:29.575 --> 00:26:29.959
right?

6c0e76d9-d2ba-43fc-8ce8-b3148551866b-0
00:26:29.960 --> 00:26:33.874
So each queue will be prefixed
with the BSQ and I'm not sure

6c0e76d9-d2ba-43fc-8ce8-b3148551866b-1
00:26:33.874 --> 00:26:37.724
what BSQ stands for but that is
how the queues are you know

6c0e76d9-d2ba-43fc-8ce8-b3148551866b-2
00:26:37.724 --> 00:26:38.880
prefixed with BSQ.

226b3e79-0935-4827-a62e-cfa8e2e4ccae-0
00:26:39.000 --> 00:26:41.000
I think it is batch step queue.

cb07a591-57a5-4117-bd4c-7f82518ff82d-0

00:26:41.000 --> 00:26:42.480
I'm not sure, I'm just guessing.

5cd4da36-0f2b-4742-bff8-5fbd6bc29913-0
00:26:42.800 --> 00:26:45.720
So let me open this in a notepad
let's see what it has.

dc841e58-e27e-4983-930c-e6f3b2170aab-0
00:26:46.040 --> 00:26:49.156
So anyway this files you cannot
read, these are not human

dc841e58-e27e-4983-930c-e6f3b2170aab-1
00:26:49.156 --> 00:26:49.640
readable.

deccb878-0f5d-4d9e-9007-8daa7c15aaf1-0
00:26:49.640 --> 00:26:52.883
So these are machine readable
and only you know mule soft

deccb878-0f5d-4d9e-9007-8daa7c15aaf1-1
00:26:52.883 --> 00:26:55.680
runtime engine can you know
understand this file.

cbacafa1-7723-4413-8a02-3181dda81eac-0
00:26:55.680 --> 00:26:56.400
So let me open.

b3fca91f-860f-496c-ade4-22514facb808-0
00:26:56.400 --> 00:26:59.480
I'm not sure which one contains
the file records.

dee2536e-a465-414d-8382-31340b768aed-0
00:27:06.150 --> 00:27:10.227
Yeah so I think yeah one queue
is empty but the other one if

dee2536e-a465-414d-8382-31340b768aed-1
00:27:10.227 --> 00:27:14.371
you see here it it it is not
human readable but you could see

dee2536e-a465-414d-8382-31340b768aed-2
00:27:14.371 --> 00:27:16.510
the records right 12345 records.

7620b914-2553-4b97-9889-5a69fd2ac238-0
00:27:16.510 --> 00:27:19.510
All the file records are
persisted using this file.

55e7cb8f-f3a7-4684-bf47-6f09367429c2-0
00:27:19.750 --> 00:27:23.787
So this is happening in the
first phase which is the load

55e7cb8f-f3a7-4684-bf47-6f09367429c2-1
00:27:23.787 --> 00:27:25.110
and dispatch phase.

54c9a45b-4412-40c4-ab41-0732cf6ba210-0

00:27:25.440 --> 00:27:28.940
So the load and dispatch phase,
it created a persistent VM queue

54c9a45b-4412-40c4-ab41-0732cf6ba210-1
00:27:28.940 --> 00:27:30.880
and persists all your fire
reports.

f5a5a065-cdf1-4cfb-80d0-eac1b380505d-0
00:27:31.120 --> 00:27:32.200
So now what happens?

c3988928-7d3a-4a34-bf83-71f6e31c0a4b-0
00:27:32.360 --> 00:27:34.600
So what is the next step which I
want to discuss?

eabbaadf-c6eb-43a0-b74a-f1546894422a-0
00:27:35.200 --> 00:27:35.600
OK.

755590f3-3216-40d0-8a52-f556a1a590ad-0
00:27:36.400 --> 00:27:37.200
And here one more thing.

d9df445a-eed6-4da4-81b8-fc12638a1fd7-0
00:27:37.200 --> 00:27:41.816
So I also told you that the
queues are mapped to a job

d9df445a-eed6-4da4-81b8-fc12638a1fd7-1
00:27:41.816 --> 00:27:43.160
instance, right?

482d6b75-93c9-4676-bda5-bf716081db78-0
00:27:43.160 --> 00:27:47.038
So in this slide I talked about
you know how the queues are you

482d6b75-93c9-4676-bda5-bf716081db78-1
00:27:47.038 --> 00:27:49.280
know, mapped to a specific
instance.

d02955a0-b004-4ebd-a852-fcb9c921e445-0
00:27:49.560 --> 00:27:55.740
So if you see the name of the
queue in the name it has a UU ID

d02955a0-b004-4ebd-a852-fcb9c921e445-1
00:27:55.740 --> 00:27:57.800
FT135 ending with 70.

c505a932-170b-4126-890d-75ce5ade38ab-0
00:27:57.800 --> 00:28:00.480
So this is nothing but the job
instance ID.

30f24388-60ec-456d-91e5-de955ba8ece6-0
00:28:00.480 --> 00:28:03.680
So if you see the logs, let me
go back.

5fb7c2a9-7218-4c96-b275-57d3a2eef1ea-0

00:28:04.480 --> 00:28:06.160
Yeah, if you see that is the
instance ID, right?

620ef406-3fa0-43d3-a0ea-346b5ab56536-0
00:28:06.160 --> 00:28:10.952
So for this instance, I mean the
queue is created with the job ID

620ef406-3fa0-43d3-a0ea-346b5ab56536-1
00:28:10.952 --> 00:28:12.840
as I mean within its name.

fb0e467a-8628-4bf3-a947-45ead1a0d3e4-0
00:28:13.000 --> 00:28:15.661
So this is how even if the
application gets stopped and

fb0e467a-8628-4bf3-a947-45ead1a0d3e4-1
00:28:15.661 --> 00:28:18.561
restarts, it is going to connect
with the same queue to, you

fb0e467a-8628-4bf3-a947-45ead1a0d3e4-2
00:28:18.561 --> 00:28:20.320
know, fetch the data and process
it.

e2eac5d1-06a3-4fd6-a229-eee04eac0b71-0
00:28:20.320 --> 00:28:24.680
So that's how the VM queues are
mapped to a job instance, OK,

e2eac5d1-06a3-4fd6-a229-eee04eac0b71-1
00:28:24.680 --> 00:28:29.040
And the same job instance ID can
also be viewed in variables.

79f69246-d2e8-440f-8bb2-aa5ffe810076-0
00:28:29.040 --> 00:28:33.072
So it's going to have a variable
created batch job instance ID

79f69246-d2e8-440f-8bb2-aa5ffe810076-1
00:28:33.072 --> 00:28:34.800
and the value is over here.

062d4f7e-251f-4409-abb7-1ea7a41f6e03-0
00:28:35.200 --> 00:28:37.920
OK, so now let's try to iterate.

d0cf1577-3ec1-4e6c-9ee5-656d80433809-0
00:28:37.920 --> 00:28:38.800
So what happens now?

660ae147-8511-4502-9880-bd0bbd85cbef-0
00:28:39.400 --> 00:28:42.905
So when you iterate this so each
record will be individually

660ae147-8511-4502-9880-bd0bbd85cbef-1
00:28:42.905 --> 00:28:43.480
processed.

f950b34c-5a8c-4405-a2af-5e11fd34021d-0

00:28:43.720 --> 00:28:47.346
So since we gave the batch block
is handed, it's going to fetch

f950b34c-5a8c-4405-a2af-5e11fd34021d-1
00:28:47.346 --> 00:28:50.803
all the file records, place it
in memory and the threads are

f950b34c-5a8c-4405-a2af-5e11fd34021d-2
00:28:50.803 --> 00:28:52.560
going to execute those records.

73ed4fe4-eb81-43d7-9f3d-ee9d15aec985-0
00:28:52.760 --> 00:28:54.720
So if you see the logger, I'm
using logger right?

3c91d236-6ffc-48d0-8540-693b112fd5ed-0
00:28:54.880 --> 00:28:59.755
So each record will be
individually logged OK And once

3c91d236-6ffc-48d0-8540-693b112fd5ed-1
00:28:59.755 --> 00:29:04.541
all the file records are
successfully processed, it's

3c91d236-6ffc-48d0-8540-693b112fd5ed-2
00:29:04.541 --> 00:29:05.960
going to say OK.

3201e3b7-9218-4abb-a880-dc7ed38da842-0
00:29:06.040 --> 00:29:10.360
So it's going to say finished
executing this instance OK.

e0e1235d-33fc-4b42-86de-ca6910cb1c42-0
00:29:10.360 --> 00:29:13.208
It's going to just close the
instance and if you go and check

e0e1235d-33fc-4b42-86de-ca6910cb1c42-1
00:29:13.208 --> 00:29:16.010
the queues, the queues will be
deleted because you completed

e0e1235d-33fc-4b42-86de-ca6910cb1c42-2
00:29:16.010 --> 00:29:17.480
processing all the fire records.

9f0d1979-62a5-49b6-9ac0-bab964477f11-0
00:29:17.480 --> 00:29:22.208
Once the processing is done, the
queues are automatically deleted

9f0d1979-62a5-49b6-9ac0-bab964477f11-1
00:29:22.208 --> 00:29:26.363
OK and and and then at the end
you also see one log being

9f0d1979-62a5-49b6-9ac0-bab964477f11-2
00:29:26.363 --> 00:29:27.080
generated.

d80c3a82-5959-493f-af7a-0d60b169e439-0
00:29:27.280 --> 00:29:31.067
So this is in this log, it is
saying that total records

d80c3a82-5959-493f-af7a-0d60b169e439-1
00:29:31.067 --> 00:29:34.720
processed, it's it's very bad,
just give me a second.

fd0a3dc2-3d41-4d8f-a819-686b7a00ce6c-0
00:29:35.520 --> 00:29:39.298
It's saying that total records
processed are 55 records were

fd0a3dc2-3d41-4d8f-a819-686b7a00ce6c-1
00:29:39.298 --> 00:29:41.280
successful and 0 records failed.

f94a3bd8-d4f8-43d7-9b3a-d5dee1e0aafc-0
00:29:41.480 --> 00:29:45.497
So this log, the last log is
provided by the on complete

f94a3bd8-d4f8-43d7-9b3a-d5dee1e0aafc-1
00:29:45.497 --> 00:29:45.920
phase.

4987fbd8-7056-4e3e-bb56-f37494f07e86-0
00:29:46.080 --> 00:29:48.655
Even if you don't provide
anything here, it's simply going

4987fbd8-7056-4e3e-bb56-f37494f07e86-1
00:29:48.655 --> 00:29:51.537
to log a message saying how many
records were processed, how many

4987fbd8-7056-4e3e-bb56-f37494f07e86-2
00:29:51.537 --> 00:29:52.760
success and how many failed.

e420cdae-4278-4aba-9d23-82d5b13b1620-0
00:29:53.320 --> 00:29:57.600
OK, so that's the first two
things which I wanted to talk.

3e573596-00ae-4f7f-b31b-c78217a939a6-0
00:29:58.080 --> 00:29:58.800
Any any questions?

32c4064b-35da-41a2-af48-fee58db928c5-0
00:29:58.800 --> 00:30:04.760
So far, no.

d4fa8264-aa27-4f49-a37d-57c5a56f2f27-0
00:30:04.760 --> 00:30:05.840
From my side, I like that.

a6a88292-5800-470b-9ebb-4973eadba36b-0
00:30:05.880 --> 00:30:07.360
Yeah, that's correct.

44607c8b-02dd-44f6-a9c3-00e3974e2748-0
00:30:07.960 --> 00:30:09.320

So let me just drag them off.

1b53b1fb-be9a-4373-98be-1590556a8780-0
00:30:09.640 --> 00:30:11.680
So the third one, so let's talk
about reliability.

2a4323f6-a4d8-4bd3-a7dc-5dc5678722ae-0
00:30:11.680 --> 00:30:12.480
So let me show you.

a4cb3da0-fb2b-4620-a364-3308f5a82da4-0
00:30:12.480 --> 00:30:15.202
Even if the application's
getting stopped, how does it

a4cb3da0-fb2b-4620-a364-3308f5a82da4-1
00:30:15.202 --> 00:30:16.440
continue processing them?

4e2c9eb0-a6c0-4943-8a23-c007f57d1e27-0
00:30:16.680 --> 00:30:20.920
OK, so for that, let me clean
the console.

ff6a29ef-b798-4d76-beca-667658c2e32c-0
00:30:21.120 --> 00:30:22.720
So let's make the same call
again.

5158de9e-20a8-47f4-aaa2-62e1b30f0119-0
00:30:24.040 --> 00:30:25.840
So sorry to interrupt you.

3d3482ff-4155-4f18-abdd-4636cb72e957-0
00:30:26.280 --> 00:30:29.636
Yeah, so it is a, it's a
endpoint Studio itself will

3d3482ff-4155-4f18-abdd-4636cb72e957-1
00:30:29.636 --> 00:30:30.080
create.

b1a5f414-1be1-4652-a903-0059f57a272b-0
00:30:30.120 --> 00:30:33.356
We don't need to explicitly
confer something on VM connect,

b1a5f414-1be1-4652-a903-0059f57a272b-1
00:30:33.356 --> 00:30:33.680
right?

3631d8bc-f2b7-40da-91dd-1ee835fcaad3-0
00:30:33.880 --> 00:30:34.240
It does.

42143e62-dd67-4fa2-bc43-e9e31f40ac82-0
00:30:34.320 --> 00:30:38.410
As soon as you drag and drop the
VM job onto that, it understands

42143e62-dd67-4fa2-bc43-e9e31f40ac82-1
00:30:38.410 --> 00:30:38.720
that.

a339f44b-1e20-4725-9218-527a0b8825b3-0
00:30:38.720 --> 00:30:42.309
So as soon as the data payload
comes to the batch job, the

a339f44b-1e20-4725-9218-527a0b8825b3-1
00:30:42.309 --> 00:30:44.560
first step is the load and
dispatch.

30001063-ab37-41b5-aa1e-924bb33da25d-0
00:30:44.560 --> 00:30:48.487
It's going to automatically,
yeah, create a VM queue versus

30001063-ab37-41b5-aa1e-924bb33da25d-1
00:30:48.487 --> 00:30:52.021
the data in there, Split the
records and process them

30001063-ab37-41b5-aa1e-924bb33da25d-2
00:30:52.021 --> 00:30:53.920
individually so that happens.

4726deba-2741-494d-8561-dc799fdaea45-0
00:30:53.920 --> 00:30:54.440
Implicit.

8d056fc0-73ea-4032-881b-ef549d901fe0-0
00:30:54.640 --> 00:30:56.280
We have no control on that.

b44fb330-ee5d-423b-90c9-2a03f6749c5f-0
00:30:57.240 --> 00:31:04.680
OK, yeah, so I'm making the same
5 calls, OK, same 5 payloads.

af9db15a-4dfe-4bae-a3b3-1786eb186db1-0
00:31:05.280 --> 00:31:09.470
So if I go back, so again, it's
going to create another job

af9db15a-4dfe-4bae-a3b3-1786eb186db1-1
00:31:09.470 --> 00:31:13.661
instance and it's going to
create another set of persistent

af9db15a-4dfe-4bae-a3b3-1786eb186db1-2
00:31:13.661 --> 00:31:14.359
VM queues.

9f1d7317-8839-439a-a076-1d8a96d0fc72-0
00:31:14.440 --> 00:31:14.840
OK.

49632d9f-d0a5-4b5e-a510-7ba4a041b990-0
00:31:15.040 --> 00:31:19.200
And now I want to show you how
reliability works, right?

5fea7fc2-648a-4478-816a-734d1015f133-0
00:31:19.200 --> 00:31:21.880
So what I want to do is I want
to process 2 records.

a9cb6c3e-c8b9-43b1-a091-31862a9c7e13-0
00:31:21.880 --> 00:31:25.080
OK, I only processed 2 records
and let's assume the application

a9cb6c3e-c8b9-43b1-a091-31862a9c7e13-1
00:31:25.080 --> 00:31:25.480
crashed.

4919bb1f-9f30-4873-a962-75ed9f763c97-0
00:31:25.720 --> 00:31:27.986
So I'm manually, you know,
closing the application here

4919bb1f-9f30-4873-a962-75ed9f763c97-1
00:31:27.986 --> 00:31:29.160
just stopping the processing.

9a5d0b7f-fc53-4046-aaaa-eb4381518256-0
00:31:29.360 --> 00:31:32.625
But you know real time scenarios
applications do crash and let me

9a5d0b7f-fc53-4046-aaaa-eb4381518256-1
00:31:32.625 --> 00:31:35.000
you know debug that again so let
me restart it.

77c4ce57-960c-4356-84eb-7e54010b22e1-0
00:31:35.200 --> 00:31:38.760
So when I restart it, the
payload is not lost.

fb97e36a-4dfc-4b53-9788-a7b8d881be78-0
00:31:38.760 --> 00:31:41.040
If you see the queues are still
available.

bab57f35-1372-471f-894a-2cdce38d01c4-0
00:31:41.480 --> 00:31:44.346
The reason the queues are still
available is because these are

bab57f35-1372-471f-894a-2cdce38d01c4-1
00:31:44.346 --> 00:31:45.120
persisted queues.

5a2be401-0835-44bc-aa19-a53da371e8f5-0
00:31:45.120 --> 00:31:48.503
Since all the records are not
yet processed, the data is still

5a2be401-0835-44bc-aa19-a53da371e8f5-1
00:31:48.503 --> 00:31:49.040
available.

904ffcc0-7765-43c9-a61b-6e7f6a6dae9f-0
00:31:49.040 --> 00:31:51.160
So you can just open this in
notepad.

0f441ef6-f84b-40dd-9f9e-6c72126b57e3-0
00:31:57.480 --> 00:31:57.840
OK.

d1032dc8-01f9-4b0d-97e0-bed0d0728371-0
00:31:58.000 --> 00:31:58.320
OK.

3477ea70-1a8c-408e-98b7-d5fe779cba2b-0
00:31:58.440 --> 00:32:03.480
So let me open this one, right.

6458abad-c321-4a56-a1ac-8f8678158046-0
00:32:03.480 --> 00:32:05.200
So the records are still
persisted, right?

c700864b-1977-4870-a8f8-02dab202ad54-0
00:32:05.200 --> 00:32:07.264
So even if the application
restarts, you don't need to

c700864b-1977-4870-a8f8-02dab202ad54-1
00:32:07.264 --> 00:32:08.240
worry about the data loss.

2735afdb-d477-4369-a511-8670635b9d3f-0
00:32:08.480 --> 00:32:12.060
So once the application comes up
running again, it's again going

2735afdb-d477-4369-a511-8670635b9d3f-1
00:32:12.060 --> 00:32:15.640
to fetch the data from the same
queue and start processing them.

98c30176-6fbb-4837-a1ab-15e283c6f3b4-0
00:32:17.560 --> 00:32:20.570
And it's the same case when you
talk about any point, I mean

98c30176-6fbb-4837-a1ab-15e283c6f3b4-1
00:32:20.570 --> 00:32:22.840
when you deploy the application
to Cloud Hub.

7d66882e-af81-440c-a85a-e4516ce99340-0
00:32:23.200 --> 00:32:27.284
So while deploying to Cloud Hub,
you need to make sure you enable

7d66882e-af81-440c-a85a-e4516ce99340-1
00:32:27.284 --> 00:32:29.760
the checkbox which says
persisted VMQS.

d3cade4c-cc5c-4b5b-974e-5c5c40bf1aff-0
00:32:30.040 --> 00:32:33.428
So then even if the application
restarts, your data will still

d3cade4c-cc5c-4b5b-974e-5c5c40bf1aff-1
00:32:33.428 --> 00:32:34.880
be persisted in Amazon SQS.

0b901efb-af31-486b-a804-073a3bd3e71d-0
00:32:35.240 --> 00:32:40.024
OK, yeah, restarting the
application is the slowest

0b901efb-af31-486b-a804-073a3bd3e71d-1
00:32:40.024 --> 00:32:40.760
process.

53b479a8-4925-4409-84d9-a8ab327b0f46-0
00:32:45.280 --> 00:32:46.640
Let's give it few seconds.

cb3a625d-a8f9-4ee7-a1c1-3d6d57fdc25d-0
00:32:53.160 --> 00:32:56.880
Yeah, Siddharth like Mark, like
a bad job process, right?

609a5c44-c2aa-42d9-a1cd-fcb715b3d054-0
00:32:56.880 --> 00:33:00.836
What we have the property, what
we do like how many jobs we can

609a5c44-c2aa-42d9-a1cd-fcb715b3d054-1
00:33:00.836 --> 00:33:04.360
define, those we can put in the
global variables, right?

add393ca-5ced-417d-b94d-baba09c67342-0
00:33:04.360 --> 00:33:08.040
That we can control from outside
this this values, right?

dcd3eeee-5f6b-4467-a77f-86dd1cc53244-0
00:33:08.400 --> 00:33:12.080
Yeah, yes, yes, of course not
variables.

895db54a-b810-4b51-a85a-1bc538a2dda9-0
00:33:12.080 --> 00:33:14.280
So those will be configuration
properties.

33e5a1b3-627c-4e9b-a811-171e2e55b92c-0
00:33:14.280 --> 00:33:18.046
So you can use config properties
and define them while you deploy

33e5a1b3-627c-4e9b-a811-171e2e55b92c-1
00:33:18.046 --> 00:33:18.960
the application.

856b52c9-3330-4230-915d-239751599d37-0
00:33:19.960 --> 00:33:20.600
OK, completely.

49309fc5-e1f6-44dd-b808-efd241594be6-0
00:33:20.600 --> 00:33:25.944
So any any any of these fields
can be you know, you know it was

49309fc5-e1f6-44dd-b808-efd241594be6-1
00:33:25.944 --> 00:33:30.955
dynamically replaced using
conflict properties, you can use

49309fc5-e1f6-44dd-b808-efd241594be6-2
00:33:30.955 --> 00:33:36.132
placeholders and you know, OK,

OK, OK, so the application got

49309fc5-e1f6-44dd-b808-efd241594be6-3
00:33:36.132 --> 00:33:41.310
restarted but now you see that
you know it's not saying batch

49309fc5-e1f6-44dd-b808-efd241594be6-4
00:33:41.310 --> 00:33:44.400
job started, it created instance
ID.

ef0aa552-7072-4b4b-ae93-7d85e10436a5-0
00:33:44.400 --> 00:33:47.504
It's not it's going to say that
because it found out that you

ef0aa552-7072-4b4b-ae93-7d85e10436a5-1
00:33:47.504 --> 00:33:50.408
know there is a previous batch
job instance which was not

ef0aa552-7072-4b4b-ae93-7d85e10436a5-2
00:33:50.408 --> 00:33:51.560
successfully completed.

b0cd446b-d442-46ff-8d05-19c44b472abd-0
00:33:51.800 --> 00:33:55.485
So if I do next, right, So I
didn't make any call in the

b0cd446b-d442-46ff-8d05-19c44b472abd-1
00:33:55.485 --> 00:33:59.364
postman, OK, So if I do next, it
is still able to fetch the

b0cd446b-d442-46ff-8d05-19c44b472abd-2
00:33:59.364 --> 00:34:02.080
previous records and process
them, right.

4637c189-7cf7-41f2-afe4-9f03e05a956f-0
00:34:02.400 --> 00:34:05.384
So this is because the queues
are persisted and this is how

4637c189-7cf7-41f2-afe4-9f03e05a956f-1
00:34:05.384 --> 00:34:08.120
the reliability you know pattern
works out-of-the-box.

08532bc6-7338-451d-b8e1-431d79300ee9-0
00:34:08.120 --> 00:34:09.480
You don't need to configure
anything.

a152eb82-eb52-4802-a3c4-f22fcf67eff2-0
00:34:09.480 --> 00:34:10.880
So this comes out-of-the-box.

249bc781-572c-4068-b685-4900c11520fa-0
00:34:12.360 --> 00:34:13.040
Makes sense.

a7e78b55-d477-43cd-bd71-5dbc43a91f88-0

00:34:14.040 --> 00:34:18.209
But OK I just some might be you
the floor for totally 5 records

a7e78b55-d477-43cd-bd71-5dbc43a91f88-1
00:34:18.209 --> 00:34:18.600
right?

bd9a1590-5106-4d5a-ab51-c44595c7a107-0
00:34:18.600 --> 00:34:21.634
But we are stopped with the
second once it's processed the

bd9a1590-5106-4d5a-ab51-c44595c7a107-1
00:34:21.634 --> 00:34:23.280
2nd it's a good good good start.

395182d7-cb14-4f3b-a2ef-8d9215f0a268-0
00:34:23.640 --> 00:34:27.167
So ideally speaking since we
stopped at two it should start

395182d7-cb14-4f3b-a2ef-8d9215f0a268-1
00:34:27.167 --> 00:34:28.520
this with 3-4 and five.

0bb87b38-fe60-4817-98a2-6581a1902e65-0
00:34:28.800 --> 00:34:32.909
But I'm really not sure why it
is processing again from the

0bb87b38-fe60-4817-98a2-6581a1902e65-1
00:34:32.909 --> 00:34:33.800
first record.

c5ee3093-601f-4c04-adfe-7344c9bc33bc-0
00:34:34.040 --> 00:34:37.640
But I think this is I'm an
unexpected even.

f4fc9986-1469-42f3-a1ff-1f462c91a37a-0
00:34:37.640 --> 00:34:38.840
I'm not sure why it happened.

1b629154-0b1c-4f2b-95ef-0c8d0ab78d45-0
00:34:39.040 --> 00:34:42.161
But when I did this earlier like
two or three months ago, it only

1b629154-0b1c-4f2b-95ef-0c8d0ab78d45-1
00:34:42.161 --> 00:34:44.999
processed from the record which
was, you know, stopped from

1b629154-0b1c-4f2b-95ef-0c8d0ab78d45-2
00:34:44.999 --> 00:34:47.080
there, it was stopped actually
three or 45.

292c58d1-b17a-41fd-8759-942a4f63971a-0
00:34:47.400 --> 00:34:49.984
And I think that would be the
same case if you use cloud as

292c58d1-b17a-41fd-8759-942a4f63971a-1

00:34:49.984 --> 00:34:50.200
well.

5735b0c8-495f-41b9-9c77-3cb3696d9980-0
00:34:50.560 --> 00:34:51.960
But I'm not sure what happened
here.

c5052b80-10e4-4baf-9ed1-e82ec8b486cb-0
00:34:52.440 --> 00:34:52.640
OK.

73537bb7-f2d0-45a8-aed9-a002260ffa1d-0
00:34:52.640 --> 00:34:56.206
So the ideal scenario is, yeah,
I think if I I think I mentioned

73537bb7-f2d0-45a8-aed9-a002260ffa1d-1
00:34:56.206 --> 00:34:56.920
here as well.

d4f5808c-8118-4ec7-b6dd-43923dc1a711-0
00:34:57.320 --> 00:35:00.200
So it's going to continue the
job execution.

f2ab1dea-ffbe-4788-b3b5-fae0df8d1368-0
00:35:00.280 --> 00:35:02.400
It's going to resume from the
point it stopped.

d8a9ae12-bad7-4f88-9a3e-4cae6548bca8-0
00:35:02.960 --> 00:35:05.569
So ideally speaking, 345 should
be processed, but now it's

d8a9ae12-bad7-4f88-9a3e-4cae6548bca8-1
00:35:05.569 --> 00:35:06.720
processing all the things.

cf12b61c-3c9b-4082-acce-043c030a4e7d-0
00:35:08.040 --> 00:35:13.120
OK, OK.

9ba50359-a3bf-479d-bfc4-0e0d3282564c-0
00:35:13.120 --> 00:35:15.200
So that's about the reliability
thing.

7a3f3db2-84be-432b-af41-8713eecd2a37-0
00:35:15.200 --> 00:35:17.741
So I'm just taking a small
example and trying to cover the

7a3f3db2-84be-432b-af41-8713eecd2a37-1
00:35:17.741 --> 00:35:18.000
topic.

125c3cda-5553-4524-8e9d-5dcc6c90483a-0
00:35:18.240 --> 00:35:20.920
So next let's talk about the
scheduling strategy.

03232460-9ad6-46bf-b968-f2dcee4362b7-0
00:35:20.920 --> 00:35:23.800

So what is this and you know how
it might help us.

bba8b72d-ae27-4dba-bfa6-1cd2b08542b2-0
00:35:24.360 --> 00:35:27.320
So if I go back, let me clean
the console.

d0e041b8-64be-44d8-ae30-33de0253dd8f-0
00:35:29.520 --> 00:35:35.323
So if you remember there is a
field called as scheduling

d0e041b8-64be-44d8-ae30-33de0253dd8f-1
00:35:35.323 --> 00:35:36.240
strategy.

a66954c0-5e3d-479a-baa5-115968ac41f2-0
00:35:36.240 --> 00:35:39.918
So the default is ordered
sequential and the other one is

a66954c0-5e3d-479a-baa5-115968ac41f2-1
00:35:39.918 --> 00:35:40.680
round Robin.

a297ce44-8d27-47a1-b0ad-9b3c499a6cca-0
00:35:40.920 --> 00:35:44.260
So first let's understand
ordered sequential and then talk

a297ce44-8d27-47a1-b0ad-9b3c499a6cca-1
00:35:44.260 --> 00:35:45.280
about round Robin.

a6a18456-7493-4087-84e5-7905bc36677c-0
00:35:45.480 --> 00:35:50.961
OK so here since we only have 5
records it's difficult to show

a6a18456-7493-4087-84e5-7905bc36677c-1
00:35:50.961 --> 00:35:52.440
you this example.

b4d7fb6c-1a50-4d8c-982a-72f7ffcc2d5f-0
00:35:52.440 --> 00:35:55.400
So what I'll do is I'll add a
wait time.

d8d346dc-123d-4205-bdf5-93be87a1dbe0-0
00:35:55.400 --> 00:35:59.300
If you remember, we can stop the
processing for X amount of time

d8d346dc-123d-4205-bdf5-93be87a1dbe0-1
00:35:59.300 --> 00:36:03.080
using a data we've function, so
I'll just use that function to

d8d346dc-123d-4205-bdf5-93be87a1dbe0-2
00:36:03.080 --> 00:36:04.520
show you how this works.

25cb66cb-3cf7-4306-89c7-85850dd9c585-0
00:36:09.760 --> 00:36:16.735

OK, so I'll just say wait, so
I'll just wait for two seconds,

25cb66cb-3cf7-4306-89c7-85850dd9c585-1
00:36:16.735 --> 00:36:18.760
no more than that.

e3fc791a-55be-45cb-aa0b-a956d8649553-0
00:36:18.760 --> 00:36:19.840
I think you have to import,
right?

b235f1c0-ed97-43ac-93ea-4bac1f4665f7-0
00:36:20.080 --> 00:36:22.800
Import the library, yeah yeah
and import yes, import.

b0da92cc-81cb-49c1-8f24-2d25ef5cea38-0
00:36:22.800 --> 00:36:24.640
Wait, yeah, that should work.

cd90e042-bdba-4fd6-a25f-6682885ec34a-0
00:36:24.920 --> 00:36:28.354
And in logger I don't want to
lock the payload, I want to show

cd90e042-bdba-4fd6-a25f-6682885ec34a-1
00:36:28.354 --> 00:36:31.952
you the job instance ID in order
to understand how this execution

cd90e042-bdba-4fd6-a25f-6682885ec34a-2
00:36:31.952 --> 00:36:32.280
works.

490fad33-b0c0-4ccb-afde-619be5ecf068-0
00:36:32.480 --> 00:36:35.160
So I want to use vas dot.

d4aa0bbe-847f-492c-8dd0-c6955151d812-0
00:36:37.240 --> 00:36:38.720
How do I get the job instance
ID?

889afbb6-2450-4b18-bf40-6d9c5746e10c-0
00:36:38.720 --> 00:36:40.520
I don't remember the variable.

fbce5941-2e06-42a0-91fd-eb8bbac2c37e-0
00:36:42.480 --> 00:36:43.680
Do you guys remember the
variable?

d133cf1a-6e02-4c70-9f78-2a2c362f94f1-0
00:36:44.280 --> 00:36:45.280
I don't remember it.

31722360-1ec3-4141-96e9-0feaa27495ff-0
00:36:48.440 --> 00:36:53.440
OK, so OK by default.

3e87928f-b2c9-447d-8cf5-18b02911214a-0
00:36:53.440 --> 00:36:54.520
OK, process variables.

7068ba4b-1a34-44b2-a44e-afa361987376-0
00:36:55.480 --> 00:36:58.840
It will be there without even we
define the variables correct?

823a07f3-13d1-4748-923f-d3c0700a5729-0
00:36:58.880 --> 00:37:00.880
Yeah, yeah it should be there.

b63a0eba-1b56-4b63-b572-ab1a822cdb0e-0
00:37:00.880 --> 00:37:03.200
But I don't remember the exact
keyword key.

bbea1b4e-13b3-4102-9c0f-2f9f346a0a84-0
00:37:03.200 --> 00:37:04.080
The key, I don't remember.

c18e6cdc-e93b-489f-88fa-769b15f89758-0
00:37:04.080 --> 00:37:07.040
So let me just make a call, go
to debug mode.

8177999f-0f87-42ac-b980-03b827665903-0
00:37:14.050 --> 00:37:16.930
Well, add a break point, go
there.

05e6adac-9bac-4cc6-9c94-d164fbd0c278-0
00:37:17.690 --> 00:37:20.250
OK, Did I use debug mode?

f4e22e83-7d75-445a-afbd-22ba5c9b2302-0
00:37:20.250 --> 00:37:21.890
I think I didn't use the debug
mode.

915aedf9-c194-4957-8b33-0ac3da533c1c-0
00:37:24.410 --> 00:37:25.410
OK, let me.

423db69e-60db-4e44-a834-3c9368027a26-0
00:37:30.560 --> 00:37:34.395
I'll use documentation because
the deployment is very very

423db69e-60db-4e44-a834-3c9368027a26-1
00:37:34.395 --> 00:37:34.720
slow.

e102f614-937a-4e09-8c3f-7cc4ae0537a6-0
00:37:36.760 --> 00:37:41.720
So let me just debug the
project, let me save it.

5cf7f4c1-62a9-4f99-b334-fe2eb500781a-0
00:37:41.720 --> 00:37:44.560
But I'll get the value new.

401d3041-9ee7-4938-9b13-11532152cbc7-0
00:38:20.360 --> 00:38:24.200
I think it is job instance ID.

a1d378b5-fab9-4c35-812e-f92ef2dba875-0
00:38:24.880 --> 00:38:25.880

Let's let's see.

8cb3a356-9edb-45ff-b374-591b9e508b31-0
00:38:27.840 --> 00:38:30.080
Yeah it says without any vars,
right?

9c25ce40-f8a7-44ef-acd1-2b608e94bd7c-0
00:38:31.040 --> 00:38:34.214
No, we can use vars in order to
refer that it it will it will be

9c25ce40-f8a7-44ef-acd1-2b608e94bd7c-1
00:38:34.214 --> 00:38:35.240
stored in a variable.

ddd667f3-c946-406f-b011-e4e27402c5cb-0
00:38:35.240 --> 00:38:38.480
So you need to use vars dot the
job instance ID.

57cb13d3-5946-4575-87c3-08371d0ff8d0-0
00:38:42.840 --> 00:38:44.120
OK, I need to wait.

7a9d7ec8-524e-4e5f-863b-c62af5a8ee89-0
00:38:53.480 --> 00:38:54.960
I think I've shown you this
earlier, right?

d1d2b77e-0a8b-4240-b606-f2175e896ece-0
00:38:54.960 --> 00:38:57.982
So in the logs it's going to
show you how many threads are

d1d2b77e-0a8b-4240-b606-f2175e896ece-1
00:38:57.982 --> 00:38:59.520
being used by the application.

e792a7da-1635-492c-8226-eb1af3fc0382-0
00:38:59.520 --> 00:39:02.925
So internally it has a formula
to decide you know how many

e792a7da-1635-492c-8226-eb1af3fc0382-1
00:39:02.925 --> 00:39:04.080
threads can be used.

93b40bc0-3fd2-4d1a-981f-0e5899a94182-0
00:39:04.080 --> 00:39:06.523
So these are the number of
threads which are available to

93b40bc0-3fd2-4d1a-981f-0e5899a94182-1
00:39:06.523 --> 00:39:07.240
this application.

b4158908-aac8-451b-bb48-dd40aaebd69a-0
00:39:07.240 --> 00:39:08.320
164 threads.

4524f3fc-e565-48e0-8154-bf82c5a0b5d3-0
00:39:08.320 --> 00:39:14.280
So did I cover this in earlier
session?

e6c66deb-2137-43c9-a0d2-34664b7fee04-0
00:39:15.600 --> 00:39:15.720
Yeah.

8fe19ab1-3556-40b6-aa35-8ab97236c18b-0
00:39:15.720 --> 00:39:18.005
One time you sold that moment
like it was just three weeks

8fe19ab1-3556-40b6-aa35-8ab97236c18b-1
00:39:18.005 --> 00:39:18.160
ago.

df46602f-9079-44f5-ba65-d423f5026b9a-0
00:39:18.360 --> 00:39:19.520
Yeah, that's it.

9a7355a2-215f-4f74-bca7-0639a55fe502-0
00:39:19.520 --> 00:39:20.880
I mean, that's the only thing I
wanted to show you.

27164360-f98b-4bd7-867e-ca9d201a10de-0
00:39:20.880 --> 00:39:21.080
Yes.

ed8517ed-868c-43ad-a4a1-854dc1f4f804-0
00:39:21.080 --> 00:39:21.400
Yes.

9c1b8990-574c-4b6a-b7f0-b8ad56275910-0
00:39:21.640 --> 00:39:25.092
So internally it uses a formula
calculating the CPU course and

9c1b8990-574c-4b6a-b7f0-b8ad56275910-1
00:39:25.092 --> 00:39:27.120
memory to come up with this
threads.

b67189ea-0ce5-4204-abfd-613620f9e9ad-0
00:39:27.240 --> 00:39:29.360
OK, yeah.

78a99405-d977-4df3-b44c-b2f83fb5064c-0
00:39:30.560 --> 00:39:31.600
OK, It is deployed.

cc3a9249-e04a-4dd2-9c99-6f87441a27b3-0
00:39:34.360 --> 00:39:35.880
Let me clean this.

7c6022cd-c3c1-42bd-ae97-ab545e950abe-0
00:39:37.600 --> 00:39:38.600
Let's make a call.

14014142-d4fd-438a-a4cc-22a8a63f37a4-0
00:39:43.520 --> 00:39:44.640
Yeah so appreciate this.

66aba7ef-3a0a-4099-a8fe-ad84c894894d-0
00:39:44.640 --> 00:39:47.040
Oh it is batch job instance ID.

7920ae6e-29d1-4465-a7ea-80dd5a5f2d98-0
00:39:47.320 --> 00:39:52.186
OK, so let me copy expression so
I can also directly copy the

7920ae6e-29d1-4465-a7ea-80dd5a5f2d98-1
00:39:52.186 --> 00:39:55.640
expression here, go to logger
and paste it.

651f317c-87de-4526-b7be-0c067ed22541-0
00:39:55.640 --> 00:39:58.080
So it's going to say bad start
job instance ID.

a1acb1cd-c1f5-4225-8a81-170c35dec794-0
00:39:58.080 --> 00:39:59.120
Even that is possible.

68a8340b-f3eb-4b12-8107-c7722ef27ddc-0
00:39:59.400 --> 00:40:01.320
OK, so let me just consume this.

35dcecdd-9494-4795-9dff-4efc769623dc-0
00:40:01.480 --> 00:40:02.320
Save this.

103eaa1f-45cc-4389-811c-bd227e60a80a-0
00:40:26.440 --> 00:40:27.480
That's going to redeploy.

8fb3af0f-a533-4e81-b566-0243cba524ff-0
00:40:30.120 --> 00:40:32.200
OK, so I introduced A2 seconds
delay.

3ac8d9cb-7e46-4f5e-86f1-3e846d40bb32-0
00:40:32.360 --> 00:40:33.600
So now let's make a call.

be9cd0f0-ae60-4937-abe5-2bbdd3f0b9f7-0
00:40:34.000 --> 00:40:35.040
So if I make a call.

a6bff2db-650d-47a4-ada0-9c0764014ce5-0
00:40:35.720 --> 00:40:38.240
So I think I should be getting
this.

ba13ab9d-8707-4b07-a0be-914cef1c028c-0
00:40:38.480 --> 00:40:40.800
OK, let me remove this break
point.

988f9db4-d16d-4ec0-9eb3-19747119c93f-0
00:40:40.800 --> 00:40:41.960
I don't want the break point.

2d2ac524-11bf-42a3-b0d6-0e28c89b0f69-0
00:40:47.910 --> 00:40:50.670
OK, so it's simply going to log
the instance ID.

b15db742-1601-46c0-96e4-06ddfa22240c-0
00:40:50.670 --> 00:40:53.150

So for this job, this is the
instance ID.

4b190939-ebf8-47c4-b4af-81724fe59cbe-0
00:40:53.430 --> 00:40:56.590
So now what I want to do is, I
just want to clean this.

fda6fca2-60e6-461d-a53b-f0bf4766c191-0
00:40:57.040 --> 00:40:59.200
I want to make two simultaneous
calls.

db5f1ea3-c699-4fdb-a697-5e1fabc0cca5-0
00:40:59.200 --> 00:41:01.680
I don't want the first job
instance to complete.

90a11318-d55c-40b8-a0f0-705bfcdf1a40-0
00:41:01.800 --> 00:41:04.883
I want to make another one while
it is executing and let's see

90a11318-d55c-40b8-a0f0-705bfcdf1a40-1
00:41:04.883 --> 00:41:05.520
what happens.

05749e0b-f1ec-4fb6-a2a7-1f70e238ccd3-0
00:41:05.760 --> 00:41:09.440
So when I make this first call,
I'm also making second call.

a6d35e88-207e-4799-a2dc-d206f97391c9-0
00:41:09.600 --> 00:41:10.880
So I made two calls.

11d47870-5adf-4f5c-982d-ebd61c66cb56-0
00:41:11.120 --> 00:41:13.480
So let's go here and see what
happens.

035e546d-2460-4919-b2a5-d640c309e146-0
00:41:13.480 --> 00:41:14.680
So let me go to console.

42177035-b51f-4708-a9ee-5383027c722e-0
00:41:17.320 --> 00:41:21.400
So it's going to log 4B6.

d8d68054-229e-4d13-8d9b-cdeb2f3037b1-0
00:41:21.400 --> 00:41:24.794
So this is I guess the first
time when I clicked the send it

d8d68054-229e-4d13-8d9b-cdeb2f3037b1-1
00:41:24.794 --> 00:41:26.520
created a job instance with 4B.

c5bb9bab-8d98-4254-ade9-d3dd2d657f34-0
00:41:26.640 --> 00:41:29.000
So it's first going to complete
the first.

686f4e17-d9cd-494f-bae8-0c17c77a08f7-0
00:41:30.480 --> 00:41:34.481

Sorry, so it's it's first going
to complete the first instance,

686f4e17-d9cd-494f-bae8-0c17c77a08f7-1
00:41:34.481 --> 00:41:35.920
the first job instance.

f8f775cc-9335-493b-bac9-ad143d1636d0-0
00:41:36.080 --> 00:41:39.483
Once the first job instance is
completed, it's going to execute

f8f775cc-9335-493b-bac9-ad143d1636d0-1
00:41:39.483 --> 00:41:40.760
the second job instance.

5357dfc6-a5e3-410c-ae9d-d26cea14821d-0
00:41:40.880 --> 00:41:45.157
So if you see the instance IDs
are different so it is 4B E it

5357dfc6-a5e3-410c-ae9d-d26cea14821d-1
00:41:45.157 --> 00:41:45.640
is 4D1.

d5d95d13-8e5d-4661-8d70-0e1c346e8b98-0
00:41:45.880 --> 00:41:47.520
So this is the default process.

e6a88964-3f54-41a7-ad5b-8c4f37beeba7-0
00:41:47.720 --> 00:41:51.902
So if one job is already running
and if you make another call,

e6a88964-3f54-41a7-ad5b-8c4f37beeba7-1
00:41:51.902 --> 00:41:56.019
it's going to create a new job
instance and the scheduling is

e6a88964-3f54-41a7-ad5b-8c4f37beeba7-2
00:41:56.019 --> 00:42:00.135
sequential, so it's going to
wait for the already running job

e6a88964-3f54-41a7-ad5b-8c4f37beeba7-3
00:42:00.135 --> 00:42:03.919
to complete and then it's going
to start the second job.

946f4c26-3c81-4db6-a4c8-7b1701aece9e-0
00:42:04.120 --> 00:42:07.000
So that is how the sequential
thing works, OK.

e49f52cd-983c-43ad-8f51-97ae0076f74b-0
00:42:07.240 --> 00:42:10.575
And if you don't want this to
happen, you have this round

e49f52cd-983c-43ad-8f51-97ae0076f74b-1
00:42:10.575 --> 00:42:10.920
Robin.

4dc9f51e-5f95-4928-83d2-8ea30bcee981-0
00:42:11.360 --> 00:42:14.903

So in round Robin it's going to
make sure that you know both the

4dc9f51e-5f95-4928-83d2-8ea30bcee981-1
00:42:14.903 --> 00:42:16.920
job instances run in parallel
again.

d451ce80-57f5-4bcb-9ceb-ff6ed7239461-0
00:42:17.720 --> 00:42:21.480
OK, so let me show you this as
well.

5f144ba1-0fee-4f4f-bc8b-d8a1d4bf01f0-0
00:42:22.120 --> 00:42:25.121
So I hope you understood this
first example where you know the

5f144ba1-0fee-4f4f-bc8b-d8a1d4bf01f0-1
00:42:25.121 --> 00:42:27.360
first job completes and the
second job starts.

baa25367-bbd6-4194-aadf-ea1e3c6d2927-0
00:42:27.600 --> 00:42:31.316
But now in round Robin both of
them will compete with the

baa25367-bbd6-4194-aadf-ea1e3c6d2927-1
00:42:31.316 --> 00:42:33.560
resources and execute in
parallel.

30f67348-b510-48b0-a3c4-771f1bf7623b-0
00:42:34.080 --> 00:42:38.315
So if you have you know very
good CPU and memory and if your

30f67348-b510-48b0-a3c4-771f1bf7623b-1
00:42:38.315 --> 00:42:42.551
application can you know, you
know handle, you know multiple

30f67348-b510-48b0-a3c4-771f1bf7623b-2
00:42:42.551 --> 00:42:46.439
job instances in parallel, you
can use the round Robin.

3ff92845-71cb-4d78-bb98-1ff75fd00821-0
00:42:47.160 --> 00:42:49.200
So let me go.

c1b458d5-f3c6-4b55-8ca6-388bb03a10f7-0
00:42:49.200 --> 00:42:52.600
So I'll make again two calls
1-2.

5755a3ac-3d12-42a4-ac67-66cf74d76e19-0
00:42:53.320 --> 00:42:54.560
OK, so let's see the logs.

936e2359-be0d-489e-ac9d-740319a3fd24-0
00:42:54.560 --> 00:43:02.854
Now I think I have to redeploy
the application and I because

936e2359-be0d-489e-ac9d-740319a3fd24-1
00:43:02.854 --> 00:43:11.148
now it's again happening in
sequence only it's not using the

936e2359-be0d-489e-ac9d-740319a3fd24-2
00:43:11.148 --> 00:43:12.780
round Robin.

06bd981e-86b6-4d54-9429-d67de3c58cf8-0
00:43:12.900 --> 00:43:14.580
So let me try this once again.

9e3f6a5d-edfb-4518-9fc1-89a7ae4254e0-0
00:43:14.620 --> 00:43:16.860
If not I think we have to
redeploy the application.

b78c5641-1218-42dc-b59b-034bb794ae41-0
00:43:17.540 --> 00:43:24.960
So let me send send or I think I
have to increase the wait time.

cf67063d-e0a3-4c4d-b65a-7411ea533ba8-0
00:43:27.080 --> 00:43:28.080
No, it's not working.

7165c699-50f6-42b9-8f0d-fd6143cb7b8e-0
00:43:28.080 --> 00:43:29.600
It's working sequentially only.

d96d7824-3b29-44d7-b7a4-e61422ab2e59-0
00:43:39.500 --> 00:43:39.700
Yeah.

a748ac29-02a5-4229-a800-2077a5abd47e-0
00:43:39.940 --> 00:43:42.069
So ideally speaking, both of
them, you know, happen in

a748ac29-02a5-4229-a800-2077a5abd47e-1
00:43:42.069 --> 00:43:43.540
parallel, but it's not happening
now.

d53ba322-0698-40b4-8d64-5847d025781d-0
00:43:44.580 --> 00:43:47.177
So they want me to stop and show
because again going to consume

d53ba322-0698-40b4-8d64-5847d025781d-1
00:43:47.177 --> 00:43:47.380
time.

176f5eda-16ae-42b3-b9b9-05915668525c-0
00:43:48.220 --> 00:43:49.660
So do you guys understood the
concept?

21b92b88-42e5-4dfe-9eb0-024608d1b542-0
00:43:50.660 --> 00:43:56.302
Yes, yes yes and the occurrence
right the occurrence you know

21b92b88-42e5-4dfe-9eb0-024608d1b542-1
00:43:56.302 --> 00:44:01.944
put the sequential and you made
occurrence as to or something

21b92b88-42e5-4dfe-9eb0-024608d1b542-2
00:44:01.944 --> 00:44:02.400
like.

f272f6e6-825d-4fb7-bdaa-e875a6f2e4c3-0
00:44:02.400 --> 00:44:04.360
Now it's by default it takes
from CPU usage.

8327ba75-ba27-43b6-99a4-d3e6fa28d4a5-0
00:44:04.400 --> 00:44:07.788
We have something called Max
occurrence, Max occurrence no

8327ba75-ba27-43b6-99a4-d3e6fa28d4a5-1
00:44:07.788 --> 00:44:10.200
this is by default now now it is
default.

27be7c42-6932-4ed7-8a2c-a55cd4f13ccd-0
00:44:10.280 --> 00:44:14.740
I think it is using 16 Max
concurrency might be 16 now by

27be7c42-6932-4ed7-8a2c-a55cd4f13ccd-1
00:44:14.740 --> 00:44:19.200
default because I have I think 8
CPU codes so 2 into 816.

9cae4f80-ddca-41c9-83ef-8e9b3771a4fd-0
00:44:19.200 --> 00:44:22.200
So it's going, it can process on
a 16 records in parallel.

56ac9bfc-d094-4928-ade7-25ad6e9dafe7-0
00:44:23.560 --> 00:44:24.000
Oh, OK.

a57986ca-e008-4da2-a737-1f68d9e7e894-0
00:44:24.000 --> 00:44:27.640
The 16 records in the job one,
basically we have two jobs.

0492e0e8-0f39-4f75-a36c-417b767cc002-0
00:44:27.640 --> 00:44:27.760
Yes.

e73f3853-21bd-4d10-91a9-739842890cd2-0
00:44:27.800 --> 00:44:28.280
We have.

ed805d90-aca6-4d08-b959-7e23d6afc698-0
00:44:28.280 --> 00:44:30.080
OK job one, it's not, we have
two jobs.

aa2f153f-f477-4435-a24c-23fb30b950e2-0
00:44:30.760 --> 00:44:31.440
It's not about.

7de0539c-c7b5-485c-89f8-1f9e23b972bb-0
00:44:32.600 --> 00:44:32.800
Yeah.

2f9c40e7-a1ba-4758-aedf-014849978b27-0
00:44:32.880 --> 00:44:33.720
For each instance.

b61419d8-56c0-45a7-ac8a-3b58badfd1ea-0
00:44:33.720 --> 00:44:34.840
This is for each instance.

604c43a8-9135-4832-9498-b11fd67cba26-0
00:44:35.600 --> 00:44:36.080
Each instance.

057137fe-1eeb-4215-a404-84b48cbe436c-0
00:44:36.080 --> 00:44:36.320
Correct.

79e401c7-f935-45b2-8534-6aae290c70c8-0
00:44:36.360 --> 00:44:36.600
OK.

0044a37e-b0a5-4c70-8d28-9e380b14f71e-0
00:44:36.680 --> 00:44:37.480
One job instance.

7da2137f-c498-49ac-b6c7-f69d98d20112-0
00:44:37.520 --> 00:44:37.640
Yeah.

3a25cfa2-503a-415e-a1fb-0031ff4a0c76-0
00:44:37.640 --> 00:44:38.200
Each instance.

dc0dce3d-dc0a-4624-ac02-ad0a1ed4af46-0
00:44:38.800 --> 00:44:39.160
Yes.

e5f722ff-bd77-40ed-aea4-f7b899c46e8b-0
00:44:39.840 --> 00:44:40.080
OK.

0235be15-ee9e-4adf-91e5-05a7db357a2f-0
00:44:40.080 --> 00:44:45.560
So that's about the second one,
the scheduling strategy.

1fc0c68c-7750-4e56-b167-8b03c6db2808-0
00:44:47.600 --> 00:44:49.160
So let's talk about 1000
records.

c99f2f4e-4dfa-4e9a-99cb-4089091c1ecd-0
00:44:49.160 --> 00:44:49.320
OK.

6b7c4972-a9da-486f-9bbc-cf65938fd278-0
00:44:49.320 --> 00:44:53.101
So let me show you how the batch
block size you know effects the

6b7c4972-a9da-486f-9bbc-cf65938fd278-1
00:44:53.101 --> 00:44:56.360

you know latency or how fast the
records get processed.

3b7f222c-9602-4dbe-a3a5-651f24c1c2fd-0
00:44:56.600 --> 00:44:58.960
So I'll switch back to
sequential.

0b46b942-6027-48c7-9af7-de4edb4c36e7-0
00:44:59.440 --> 00:45:03.251
So first I would like to use
batch block size as 10 and I

0b46b942-6027-48c7-9af7-de4edb4c36e7-1
00:45:03.251 --> 00:45:05.880
would like to process thousand
records.

de35f2d5-e37c-4653-9d83-b547f3aed488-0
00:45:05.920 --> 00:45:07.800
OK not 5 records.

8c2401a9-0af7-4ec9-9e1f-398b415c1dba-0
00:45:08.000 --> 00:45:12.219
So what I have is I have a CSV
file so I want to read the CSV

8c2401a9-0af7-4ec9-9e1f-398b415c1dba-1
00:45:12.219 --> 00:45:12.560
file.

6411c12c-9134-4db4-860c-6ba2918e3d7a-0
00:45:12.560 --> 00:45:14.880
Let me use the file connector.

1fa59b85-a81c-4b2a-afb3-e51c07a325d7-0
00:45:15.800 --> 00:45:18.120
I I just want to use the read
operation.

e49a200f-ae92-4804-bf8a-edc8a64d3539-0
00:45:18.920 --> 00:45:22.640
OK, I want to pass the CSV file
using this file.

215c1c07-88ed-4758-80ae-c8dfe8675ee3-0
00:45:23.080 --> 00:45:30.846
I think the CSV file is running
or it's there in my local to go

215c1c07-88ed-4758-80ae-c8dfe8675ee3-1
00:45:30.846 --> 00:45:33.880
to C directory CSV right?

94095ab7-a222-4497-945c-c44f75ddc1a0-0
00:45:33.880 --> 00:45:38.400
So let me open this and
notepad++.

9f31961d-5ac3-49dc-9c74-8b76c2bf500c-0
00:45:39.240 --> 00:45:40.680
I can't drag and drop stuff.

52cf1d66-e47c-4fab-899e-2674c0542a7b-0
00:45:41.720 --> 00:45:43.720

Windows is very weird few times.

4aca1e36-23a1-472a-8548-eff8e6591bdf-0
00:45:45.520 --> 00:45:49.720
OK, so if you see it has 1000
records, right?

684206c4-0244-406e-b377-a523bab80f0f-0
00:45:49.720 --> 00:45:51.640
Thousand I think thousand more
than 1000 records.

aea538b6-3273-4d4d-ae2f-59bccdcd287c-0
00:45:51.640 --> 00:45:54.360
OK, so there are some repeated
records but that's fine.

d9011b68-5ae1-47b7-b9ba-6efc3bb39bb9-0
00:45:54.520 --> 00:45:57.000
So I want to use the read
operation.

4e8b997a-9ab2-4beb-a0aa-cf9ad4d99205-0
00:45:57.280 --> 00:46:02.040
So let me say C it's in C in my
local directory users dot CSV.

06da6507-efee-4e07-a6a1-2764911ba520-0
00:46:02.440 --> 00:46:04.600
I want to remove this transform
message.

28c6aeaf-50cd-4d18-becd-fcb51be3d9d0-0
00:46:04.600 --> 00:46:06.240
I don't want any logger as well.

6b548622-af9b-471a-b3b1-8eb8dcfa25de-0
00:46:06.400 --> 00:46:09.660
I want to show you how the
processing effects based on the

6b548622-af9b-471a-b3b1-8eb8dcfa25de-1
00:46:09.660 --> 00:46:10.600
batch block size.

7b4a716a-f3d2-4202-bc7f-7c46f94f42c2-0
00:46:10.920 --> 00:46:15.760
So I just want to have a simple
set payload.

60992ec9-febf-4a27-83fe-e2691df63057-0
00:46:15.760 --> 00:46:20.403
I don't want to set any payload,
you know, I just want to have,

60992ec9-febf-4a27-83fe-e2691df63057-1
00:46:20.403 --> 00:46:22.000
you know, set payload.

8e602442-3a14-4dd7-90a6-f7570445ea4f-0
00:46:22.080 --> 00:46:25.652
OK, so if you I mean the only
thing which you have to remember

8e602442-3a14-4dd7-90a6-f7570445ea4f-1

00:46:25.652 --> 00:46:27.240
is I'm using block sizes 10.

45a71cf0-ce71-45c2-8a07-77ef03446ca6-0
00:46:27.560 --> 00:46:31.200
So now we have no let me, I mean
it's saving.

aa79ee9b-84ae-4671-8a3c-aa17a7002e9a-0
00:46:31.200 --> 00:46:32.600
The application will be
redeployed.

87d77ac1-0dee-4124-8731-3ec34ec59a98-0
00:46:33.000 --> 00:46:38.572
So if I go back to my PPT for a
second, if you remember this

87d77ac1-0dee-4124-8731-3ec34ec59a98-1
00:46:38.572 --> 00:46:43.688
slide, I have 1000 records in
the CSV and this thousand

87d77ac1-0dee-4124-8731-3ec34ec59a98-2
00:46:43.688 --> 00:46:47.160
records will be persisted in the
VMQ.

421479d1-ee6c-4787-a744-569bcfe5e08e-0
00:46:47.800 --> 00:46:50.280
OK, let me add a break point
here.

678c4363-d40e-4783-8b53-1dbef624fe4a-0
00:46:50.280 --> 00:46:51.840
OK, no need of any break points.

7cc595db-a7c9-42ae-9150-3749b0172663-0
00:46:52.400 --> 00:46:52.760
OK, sorry.

2afd6479-7c25-451b-a896-57a7bc8ef8a4-0
00:46:52.960 --> 00:46:56.545
Yeah, so the when the call gets
executed, the batch job is going

2afd6479-7c25-451b-a896-57a7bc8ef8a4-1
00:46:56.545 --> 00:47:00.020
to create a VMQ and persist to
all the 1000 records and I give

2afd6479-7c25-451b-a896-57a7bc8ef8a4-2
00:47:00.020 --> 00:47:01.400
the batch block sizes 10.

59a57cbd-89c1-4cd0-a23f-a44a8a06232b-0
00:47:01.400 --> 00:47:04.980
So when I say 10, it's going to
fetch 10 records from the VMQ

59a57cbd-89c1-4cd0-a23f-a44a8a06232b-1
00:47:04.980 --> 00:47:08.099
and store them in memory and
those 10 records will be

59a57cbd-89c1-4cd0-a23f-a44a8a06232b-2
00:47:08.099 --> 00:47:11.565
processed in step one, again
inserted to the queue and then

59a57cbd-89c1-4cd0-a23f-a44a8a06232b-3
00:47:11.565 --> 00:47:14.280
it's going to fetch another
different records.

5943d071-1e4b-47aa-87c9-5801ec071ba9-0
00:47:14.320 --> 00:47:15.080
That's the processing.

0b51a522-4576-4bec-be1a-a17df525cd92-0
00:47:15.080 --> 00:47:16.200
So that's how it's going to do.

214d4f4f-306a-44c5-9088-1cf544ce3905-0
00:47:16.360 --> 00:47:21.257
So since we have 1000 records
and the block size is 10, it has

214d4f4f-306a-44c5-9088-1cf544ce3905-1
00:47:21.257 --> 00:47:25.998
to, you know, fetch, return,
fetch return almost like 200 or

214d4f4f-306a-44c5-9088-1cf544ce3905-2
00:47:25.998 --> 00:47:27.320
300 times, right?

cf581313-88e6-4fa0-bd6f-df2186c8db95-0
00:47:27.320 --> 00:47:30.463
So because each fetch can only
fetch 10 records at a time, it

cf581313-88e6-4fa0-bd6f-df2186c8db95-1
00:47:30.463 --> 00:47:33.506
has to fetch and again, you
know, send back, fetch and send

cf581313-88e6-4fa0-bd6f-df2186c8db95-2
00:47:33.506 --> 00:47:33.760
back.

f9582450-f438-4987-8482-55b4279c339c-0
00:47:33.920 --> 00:47:35.160
OK, so let's see.

dfd98587-27fc-418f-b0f3-a4f9836f2c34-0
00:47:35.160 --> 00:47:35.280
I'm.

bffbaa80-a029-4f3d-8cc6-28125fc01b68-0
00:47:36.640 --> 00:47:39.280
I'm sorry, 100 * 100 * 100
times.

49e7aecc-3382-44f0-8a6d-967b5d7424dc-0
00:47:39.280 --> 00:47:40.120
Yeah, 100 times.

6d0fbafd-ce8c-4a00-9f9c-ab1f549dab98-0
00:47:40.120 --> 00:47:40.640

Sorry, yes.

2b2caf54-d4fd-444e-95aa-d3810e43e761-0
00:47:40.800 --> 00:47:44.280
And again this all are like
input, output operations.

72d66d20-a2eb-4b7c-b4ac-25cc44f9f4ff-0
00:47:44.280 --> 00:47:47.994
It has to input and output,
input, output 100 times and now

72d66d20-a2eb-4b7c-b4ac-25cc44f9f4ff-1
00:47:47.994 --> 00:47:49.480
we are doing it in file.

8c67f06c-6e3e-4815-8925-c0dd80022b91-0
00:47:49.480 --> 00:47:52.156
But if you go to cloud up, it
has to connect to an external

8c67f06c-6e3e-4815-8925-c0dd80022b91-1
00:47:52.156 --> 00:47:54.120
Amazon SQS and do the pull in
and pull out.

b4dfb2d2-bb2b-473c-9f73-6dda8afcd3fa-0
00:47:54.120 --> 00:47:55.120
I mean input and output.

af1b6171-f642-4913-a19c-8f77b3149260-0
00:47:55.120 --> 00:47:57.880
So it's going to, it's it's a
very costly operation.

0471f0e0-14c9-4806-90d8-5f922f64459e-0
00:47:57.880 --> 00:48:00.947
So you have to make sure that
the block size is a good number

0471f0e0-14c9-4806-90d8-5f922f64459e-1
00:48:00.947 --> 00:48:03.520
based on the number of records
you want to process.

6c61dcec-0b09-4c1a-a3d9-28958824f124-0
00:48:03.520 --> 00:48:06.450
So you have to do a lot of
testing and and come up with a,

6c61dcec-0b09-4c1a-a3d9-28958824f124-1
00:48:06.450 --> 00:48:09.330
you know, block size which is
suitable you know, for your

6c61dcec-0b09-4c1a-a3d9-28958824f124-2
00:48:09.330 --> 00:48:10.919
application or for your records.

0acb18a8-a0e4-4b15-a430-b0eb8c472509-0
00:48:11.360 --> 00:48:12.320
So let me show you why.

5db5f144-007c-45b9-b9f1-ae464ce2fb14-0
00:48:12.720 --> 00:48:14.080

So I think it is deployed.

a4d77e79-e64b-4ea2-bf90-01cf2d0d928e-0
00:48:16.040 --> 00:48:16.960
Yeah, let me clean it.

b8ff4360-f8c1-4695-a0cb-491a05f0666c-0
00:48:18.680 --> 00:48:20.920
So let's just make a call.

dc480c45-e447-4a06-a40b-de4c2724800c-0
00:48:20.960 --> 00:48:23.160
So instead of POST, I'll just
make a GET call.

51e5026e-9559-4818-8107-cd040f2e6ec4-0
00:48:23.160 --> 00:48:25.200
OK, it's an array, but for it to
take is it?

0322ef91-8195-4ae1-b4a5-3b18dfbd4408-0
00:48:25.560 --> 00:48:28.200
I'm just asking yes, yes, yes,
no.

599f8bd5-cb54-4e03-88ee-bdcb5fe3aa75-0
00:48:28.200 --> 00:48:30.960
CSV takes as array format.

341d633c-0868-4633-9f7b-1ad019acf17e-0
00:48:31.560 --> 00:48:32.320
It's going to convert.

c9b33461-65ad-4229-82b0-2b97642d90d9-0
00:48:32.320 --> 00:48:32.600
Yes.

141cff87-185e-49a5-a693-fb7f8882bca4-0
00:48:32.600 --> 00:48:32.920
Array.

9809d71e-03c3-4286-89e1-589a3e3433f0-0
00:48:32.920 --> 00:48:33.840
It's going to convert in a
array.

92325acf-f9de-40dc-a118-5aaaec9e7a6f-0
00:48:33.840 --> 00:48:36.400
But we didn't write any data
view code or anything like that.

7e431365-18f3-473d-bdd7-3d55c022ff28-0
00:48:36.400 --> 00:48:37.240
You don't need to write.

7fb0fe24-f9c9-42f4-a0da-96cac93f572e-0
00:48:37.400 --> 00:48:38.000
You don't need.

00f80825-474b-4028-b1d8-e5400b303432-0
00:48:38.000 --> 00:48:38.720
You don't need to write.

a6a1647b-5e97-4ce9-a386-ad4bbd972771-0
00:48:38.920 --> 00:48:39.480

Yeah it's.

22b95c54-2b4f-4f77-9c06-91ebca28bf09-0
00:48:39.760 --> 00:48:41.080
So if you say in CSV, right.

0144f7dc-6729-492c-a794-7facf1e07f82-0
00:48:41.080 --> 00:48:43.640
So each record will be stored
separately.

7cc3388d-87e1-4e09-8148-0653b6e654d3-0
00:48:43.640 --> 00:48:45.760
It's going to treat each but as
a separate.

739aa90e-27a9-4d4f-b698-1704fbe6c38d-0
00:48:45.760 --> 00:48:47.240
Just no.

9d8f9448-aea2-42ad-8e21-c0df0b193e53-0
00:48:47.240 --> 00:48:48.560
But you don't need to do
anything.

ffd209a8-c2ab-411c-9011-5b204285cb82-0
00:48:48.880 --> 00:48:49.720
OK, OK, fine.

8e4b0517-81e9-493e-8bb8-393a42579b47-0
00:48:49.720 --> 00:48:49.840
Fine.

fe8ba428-af67-40a2-9f93-ba5267ac0a0e-0
00:48:50.400 --> 00:48:53.240
So if you let me show you the
documentation.

bf540efc-0a95-4585-ae08-1658e2fd8c4c-0
00:48:53.240 --> 00:48:53.840
So it's killed.

bd6e297a-c3eb-41f4-bc29-2b6ae99ae983-0
00:48:53.960 --> 00:48:57.500
Clearly mentioned in the
documentation saying that if I

bd6e297a-c3eb-41f4-bc29-2b6ae99ae983-1
00:48:57.500 --> 00:48:57.880
go to.

dc1191a6-950c-41b3-9e4c-31b0bfd3fbd6-0
00:49:00.680 --> 00:49:02.520
Yeah, batch job phases.

adf28e0f-8c69-418b-9e12-1810122d78ea-0
00:49:03.080 --> 00:49:05.800
There is something called S what
is the valid input?

553c34de-3105-447b-8bbb-3241028fcfee-0
00:49:06.120 --> 00:49:11.443
So the valid input for batch job
is Java iterated arrays as well

553c34de-3105-447b-8bbb-3241028fcfee-1
00:49:11.443 --> 00:49:14.720
as Jason XML and also CSML also,
right?

74429b6b-314f-4e3e-94db-40dcd2e38f07-0
00:49:14.720 --> 00:49:15.320
Oh, XML.

916697e2-c272-43d9-a5db-a52d6b1441bb-0
00:49:15.320 --> 00:49:15.840
Also XML.

e757ddb1-45f7-4de3-bca6-abf6f2110907-0
00:49:16.200 --> 00:49:17.000
Even XML.

0f6efcbf-3dad-4ba3-abd1-9da9336104f7-0
00:49:17.000 --> 00:49:18.320
Yeah, OK.

70e65c0b-d73e-4a99-8113-53e2b58f1bae-0
00:49:19.600 --> 00:49:19.960
OK.

98d8c32b-99ac-45f1-a610-cfbc2682b2fb-0
00:49:20.200 --> 00:49:23.480
So yeah, I'll need to make a
call.

fa832296-0f80-4a08-be8a-82154f4f5de8-0
00:49:23.560 --> 00:49:24.720
So let's make a call.

1d54ec80-ca9a-4c24-a55c-58c8629c1ee2-0
00:49:27.240 --> 00:49:30.225
I'm not using any debug point,
so it has to simply, you know,

1d54ec80-ca9a-4c24-a55c-58c8629c1ee2-1
00:49:30.225 --> 00:49:32.440
continue processing and show me
all the logs.

5b05cc21-ca4e-48b9-8ab1-6d00fb5c6b2d-0
00:49:32.720 --> 00:49:33.960
So let's check the logs now.

208e1347-1ff8-417a-9c53-523f2452951c-0
00:49:33.960 --> 00:49:37.587
So I want to show you what would
happen if you use block sizes

208e1347-1ff8-417a-9c53-523f2452951c-1
00:49:37.587 --> 00:49:37.760
10.

1a84b95c-9181-4378-a5d9-1ac850d27abe-0
00:49:38.080 --> 00:49:41.757
So it's saying it created an
instance, started loading, and

1a84b95c-9181-4378-a5d9-1ac850d27abe-1
00:49:41.757 --> 00:49:42.800
finished loading.

2d339320-55c8-4c6e-b739-e650178da211-0
00:49:42.800 --> 00:49:45.873
In the finished loading, you can
see that thousand run records

2d339320-55c8-4c6e-b739-e650178da211-1
00:49:45.873 --> 00:49:47.240
were loaded in the VM queue.

4a0067fd-d745-4ba9-9a37-f809d015ad12-0
00:49:47.240 --> 00:49:51.757
So if you go, a VM queue would
be created and then it is saying

4a0067fd-d745-4ba9-9a37-f809d015ad12-1
00:49:51.757 --> 00:49:53.240
it started execution.

156939b1-36f0-4145-be20-3465e7da15ad-0
00:49:53.440 --> 00:49:58.000
So now I want to no at what time
the execution started.

13f15e74-cd6a-4abf-b91f-408b70dd71db-0
00:49:59.560 --> 00:50:03.840
So the execution started at
184950.

90ecc9b9-334d-4a83-be16-66b5f2da93f0-0
00:50:04.040 --> 00:50:06.697
So I mean so let me just note
down this number and show you

90ecc9b9-334d-4a83-be16-66b5f2da93f0-1
00:50:06.697 --> 00:50:09.399
what this numbers are like I
just want to calculate how much

90ecc9b9-334d-4a83-be16-66b5f2da93f0-2
00:50:09.399 --> 00:50:10.640
time it took for processing.

201f7d16-0501-4c2b-b8b0-f4546238c16d-0
00:50:10.880 --> 00:50:11.520
So it is I'm.

b58d98a5-5f44-4ed7-8dcd-d79becd240f1-0
00:50:11.520 --> 00:50:12.880
I'm just going to take the
seconds.

b4ba145a-d1d2-4920-8ce3-84b4fc3546cd-0
00:50:12.880 --> 00:50:16.000
OK, it took seconds to complete
so it is 50.

9bf23ccd-be28-4241-bcb9-5ea8537b94b8-0
00:50:16.000 --> 00:50:22.600
So it's it's like 5 3.8, I mean
3.8.

7de65601-593c-485b-841a-0fac996cfc9b-0
00:50:22.600 --> 00:50:26.784
But yeah, I just want to make

sure 3822 even milliseconds I

7de65601-593c-485b-841a-0fac996cfc9b-1
00:50:26.784 --> 00:50:28.040
want to calculate.

3a3ab1a7-0946-40d0-a169-3dc06887de88-0
00:50:28.440 --> 00:50:32.485
So I took around 3817
milliseconds, so 3.8 seconds to

3a3ab1a7-0946-40d0-a169-3dc06887de88-1
00:50:32.485 --> 00:50:37.055
process all the 1000 records
because the batch block size is

3a3ab1a7-0946-40d0-a169-3dc06887de88-2
00:50:37.055 --> 00:50:37.280
10.

8869a0f6-4688-4779-85c0-4a833ff21073-0
00:50:37.640 --> 00:50:42.482
So now let's increase the block
size, so let's use 500 and see

8869a0f6-4688-4779-85c0-4a833ff21073-1
00:50:42.482 --> 00:50:45.480
how much time it takes for
processing.

2f6deb18-3294-4111-8e9c-4eab3d32cf0c-0
00:50:47.960 --> 00:50:51.555
So you can also use 1000 as
well, but you need to also

2f6deb18-3294-4111-8e9c-4eab3d32cf0c-1
00:50:51.555 --> 00:50:54.040
understand what are the
implications.

73076d88-a825-47fd-98a6-844287e55ccb-0
00:50:54.200 --> 00:50:58.201
So when you use 500, if you go
back to the slide, it's going to

73076d88-a825-47fd-98a6-844287e55ccb-1
00:50:58.201 --> 00:51:00.640
fetch 500 records and store in
memory.

d0c94818-1372-4620-baa3-3c134cf62134-0
00:51:00.920 --> 00:51:03.912
If your application does not
have the RAM, it's going to

d0c94818-1372-4620-baa3-3c134cf62134-1
00:51:03.912 --> 00:51:05.120
crash your application.

9a5e6d43-73ab-421a-a4f5-d892c62c3e62-0
00:51:05.120 --> 00:51:08.096
So you have to do a lot of
testing and then come up with

9a5e6d43-73ab-421a-a4f5-d892c62c3e62-1

00:51:08.096 --> 00:51:08.880
the block size.

6b5c9b66-8a2a-4594-9942-547228aa51fe-0
00:51:09.120 --> 00:51:13.102
OK, so yeah, so that's how you
need to calculate the block

6b5c9b66-8a2a-4594-9942-547228aa51fe-1
00:51:13.102 --> 00:51:13.440
size.

d9528e94-23ea-423c-b027-7ff30f17d358-0
00:51:13.440 --> 00:51:13.680
OK.

bf27265c-908b-4a23-9270-a1d82c661153-0
00:51:13.960 --> 00:51:18.076
But I think I'm just using a
very small data set and I know

bf27265c-908b-4a23-9270-a1d82c661153-1
00:51:18.076 --> 00:51:22.193
my RAM can handle 500 records,
so I'm just using 500 as the

bf27265c-908b-4a23-9270-a1d82c661153-2
00:51:22.193 --> 00:51:23.360
batch block size.

2f459c2e-0373-45f9-a13f-8c6e0599bb11-0
00:51:23.440 --> 00:51:25.760
OK, so let me go to console.

62b10aa6-1004-4686-b319-e2ef2fd944a2-0
00:51:25.760 --> 00:51:29.321
It is redeployed, so let's make
the same call again with the

62b10aa6-1004-4686-b319-e2ef2fd944a2-1
00:51:29.321 --> 00:51:30.080
1000 records.

9d42b14f-627b-4e3b-a2b2-dceef90f4dfc-0
00:51:34.960 --> 00:51:36.200
Yeah, so it completed.

849ec538-21db-4d68-8367-6ae1353a4f59-0
00:51:36.200 --> 00:51:37.960
I think it's far quicker now.

909f8994-8c89-424e-96b2-d036d4c4e84a-0
00:51:37.960 --> 00:51:40.360
So when did it start?

a054b6dc-43c9-421b-a481-dfc1478e0df3-0
00:51:40.360 --> 00:51:46.320
So if I see this, it started at
5155 and edit it 5156.

0b1e8b2a-304f-41d1-bf97-b7446b278c8f-0
00:51:46.320 --> 00:51:49.840
I think it's around one second,
not even 2 seconds.

b7004913-095a-4458-9138-3982cc684032-0
00:51:50.400 --> 00:51:59.510
So if I go with the fiftyfive
0711.12, yeah, something 56236,

b7004913-095a-4458-9138-3982cc684032-1
00:51:59.510 --> 00:52:04.800
yeah, so it took around 1.1
second.

17f2a3ef-f23c-4139-98d1-b8ecd4b555aa-0
00:52:04.880 --> 00:52:07.797
So here itself you can see that
there is a difference about more

17f2a3ef-f23c-4139-98d1-b8ecd4b555aa-1
00:52:07.797 --> 00:52:08.560
than 2.5 seconds.

6ff7ffb3-c492-4774-be53-56ea79cbd043-0
00:52:08.560 --> 00:52:11.440
And this is just for 500 records
or just for 1000 records.

ae4c75f8-0b6f-4020-8cba-f49b1e416c9a-0
00:52:11.440 --> 00:52:14.531
So just assume if you're using a
million records, so you could

ae4c75f8-0b6f-4020-8cba-f49b1e416c9a-1
00:52:14.531 --> 00:52:16.200
save a lot of time in
processing.

8f0de122-9836-41e5-aa74-5e6af14c124d-0
00:52:16.240 --> 00:52:16.520
OK.

980a5b1a-a43a-4cc6-8d88-3d8738f30f66-0
00:52:16.760 --> 00:52:19.673
But this number make sure that
you know, you have, you know,

980a5b1a-a43a-4cc6-8d88-3d8738f30f66-1
00:52:19.673 --> 00:52:22.635
calculated how much memory you
have, what is the size of your

980a5b1a-a43a-4cc6-8d88-3d8738f30f66-2
00:52:22.635 --> 00:52:23.399
payloads, right.

66c8de43-17eb-426c-b2b8-ab465fea75bf-0
00:52:23.400 --> 00:52:26.360
So based on that, the batch size
block has to be decided.

b7e6f33f-ec52-4e92-859b-7543cc771965-0
00:52:27.760 --> 00:52:28.040
Yep.

e08ee360-dcdd-4530-845b-ded88f29806f-0
00:52:28.120 --> 00:52:38.800
Any any questions so far, I
think we're good.

e199d864-70d1-410e-95ef-d39a38f31128-0
00:52:39.000 --> 00:52:39.920
Yeah, good.

465be532-2960-47df-8248-fb1f87cbdf49-0
00:52:40.040 --> 00:52:40.280
Right.

e21d0303-5d3c-49de-a437-e66388bd413f-0
00:52:40.280 --> 00:52:40.960
So let's move on.

badf47f2-f370-4184-a7cb-370fb7468afa-0
00:52:41.000 --> 00:52:42.640
So we have few more topics.

c4a0db2e-00b4-447f-8bd6-82446c01d13c-0
00:52:42.640 --> 00:52:45.440
So now we'll be working with
multiple batch steps.

f6210644-b203-4258-9351-13a2b13363d9-0
00:52:45.440 --> 00:52:45.680
OK.

f222e79e-a8f2-4518-b75e-7bd1248e414b-0
00:52:45.680 --> 00:52:46.600
So instead of 1.

765f7880-58f8-4cfc-9087-7a5fa1562335-0
00:52:46.600 --> 00:52:51.279
So I think I covered so one
that's that's yeah, so you said

765f7880-58f8-4cfc-9087-7a5fa1562335-1
00:52:51.279 --> 00:52:56.114
some XML like I think we can
actually have a, so you have put

765f7880-58f8-4cfc-9087-7a5fa1562335-2
00:52:56.114 --> 00:52:57.440
CSV right now on.

1a720574-25d4-483e-865f-8538bbf5bda2-0
00:52:58.200 --> 00:53:00.320
I think that operations might be
possible to do.

738d248a-80a1-4bcb-b9e1-4cbaa1e07bfd-0
00:53:00.320 --> 00:53:01.480
Ignore the root.

b3a1d5a4-87c8-4bf6-b361-4cd9eec008a3-0
00:53:01.520 --> 00:53:02.880
You have root elements and all
those things.

75300185-64a5-400f-8866-91e805e3f924-0
00:53:02.880 --> 00:53:03.440
You can take it up.

a67c7ca5-2e1c-47dd-9199-d5df63be9a44-0
00:53:03.960 --> 00:53:04.160
Awesome.

04ae04e9-7481-437c-a191-12e90389b780-0
00:53:04.160 --> 00:53:06.200
I think the route will be
automatically ignored.

91cfd7c0-0e54-4106-9637-0e9d5963663b-0
00:53:06.200 --> 00:53:07.360
It won't store the route.

ac283c4c-f8ea-4ec8-b85f-53ff6b9dbdd7-0
00:53:07.880 --> 00:53:11.000
OK, so in XML, I mean CSV
especially, it's going to

ac283c4c-f8ea-4ec8-b85f-53ff6b9dbdd7-1
00:53:11.000 --> 00:53:14.840
automatically identify that it's
a CSV and it's going to ignore

ac283c4c-f8ea-4ec8-b85f-53ff6b9dbdd7-2
00:53:14.840 --> 00:53:16.880
the the route one, the first
one.

822d4d7d-610c-415a-9488-4d1a43c35471-0
00:53:17.120 --> 00:53:20.040
But in XML, I haven't tried with
XML, honestly speaking.

9badb0ec-2c95-4e2b-931f-2d3f32898585-0
00:53:20.160 --> 00:53:21.000
OK, so I'm not showing.

6cac4a33-6579-46ea-a957-d2e89fe73e65-0
00:53:21.760 --> 00:53:22.720
OK, ignore the first.

49cab89a-a47c-4229-80f6-d97621fdd23d-0
00:53:22.840 --> 00:53:23.640
OK, the first.

01c333f1-b8f7-456d-829e-69bc8143d726-0
00:53:23.640 --> 00:53:24.400
No, it's here.

6d5e60a4-af4e-4a7e-954c-6faaac74349d-0
00:53:24.400 --> 00:53:24.880
It's header.

8ee3b37f-811a-4545-8bb8-e6a074c4c8d1-0
00:53:24.920 --> 00:53:25.520
I mean, sorry.

f7d94d44-bef0-4353-a80a-63850142d134-0
00:53:25.520 --> 00:53:27.240
In CSV it's gonna ignore the
header part.

38bc5a14-70ad-4c4e-b385-b0098321cd44-0
00:53:27.360 --> 00:53:27.760
Header.

6d08ed8c-453c-49f0-8657-267a9bf0906e-0
00:53:27.960 --> 00:53:28.120

Yeah.

e613ab84-7376-4c73-9976-f351f2cc80c4-0
00:53:28.120 --> 00:53:29.520
Yeah, it should define I think.

d43f98a3-d98f-4160-bdc1-ea1af95348ba-0
00:53:29.880 --> 00:53:33.256
I think the read activity you
have to define separately I

d43f98a3-d98f-4160-bdc1-ea1af95348ba-1
00:53:33.256 --> 00:53:35.760
think I I don't know I'm just
guessing it.

fe8c3931-9c38-4776-8f09-e6eeff3fc537-0
00:53:35.760 --> 00:53:35.880
OK.

390fdd2e-f916-4c75-b25a-65f7a60b2e02-0
00:53:35.880 --> 00:53:37.200
I'm not sure on the XML part.

93b01ca3-33b1-45c8-9870-a622b745ba79-0
00:53:37.680 --> 00:53:38.720
Not XML here.

22696022-5a56-47dd-a398-d88cbfd75491-0
00:53:38.720 --> 00:53:41.240
Also CSV you have a reader what
is it called?

bbee00ca-5637-421c-be69-9d95889d3b49-0
00:53:41.240 --> 00:53:42.520
Connector or whatever read.

e04ed3aa-e66d-4650-a7e1-432a6f568e14-0
00:53:42.640 --> 00:53:46.400
So this is just fetching,
fetching the C CSV.

ad5f1396-e28f-443d-a92d-600135aa05b9-0
00:53:46.400 --> 00:53:46.920
That's it.

02bf8db4-9323-4e9f-af07-439f84699851-0
00:53:47.440 --> 00:53:50.960
You can also use a set payload
and copy paste the entire CSV.

a6942cca-4fe6-406a-a418-a7f916adc8ab-0
00:53:51.240 --> 00:53:51.960
It's the same thing.

b5553bce-e092-414b-92b5-6f7eda3d2a8d-0
00:53:52.080 --> 00:53:54.000
So just to show you that I can
use a file.

a2e7da9c-013e-402b-a117-752c30d76491-0
00:53:54.240 --> 00:53:56.560
I'm using the read connector the
it's the same thing.

bea960b1-302c-4306-8ddc-659689b9d35d-0
00:53:56.720 --> 00:53:56.840
Yeah.

42b55565-ef72-4dac-b279-fdfd27cd980b-0
00:53:59.040 --> 00:54:01.320
So now let's go with our
example.

7cd2cce2-4f4f-4a38-8990-3f9510ed1e12-0
00:54:01.320 --> 00:54:02.120
So we have.

f6278e87-98bf-462e-88ca-48e828c847d8-0
00:54:02.120 --> 00:54:04.080
I have a different example for
the next steps.

13794edd-ca75-4d48-85c3-dccabc40224b-0
00:54:04.360 --> 00:54:09.200
So I want to use you know this
10 records.

a6542a4d-81fa-4830-8cb4-2581ce768471-0
00:54:09.200 --> 00:54:11.120
OK, I want to process this 10
records.

5b7a8b52-f6b5-4d1b-a30d-ee19ef0015b6-0
00:54:11.320 --> 00:54:16.037
If you see in the 10 records I
have a name, country, e-mail and

5b7a8b52-f6b5-4d1b-a30d-ee19ef0015b6-1
00:54:16.037 --> 00:54:19.280
ID and ID is starting from
123456 up to 10.

d7dcacdb-9ced-42a9-abbf-31a7c1d0f901-0
00:54:19.640 --> 00:54:23.627
So what I want to do in batch
processing is I only want to

d7dcacdb-9ced-42a9-abbf-31a7c1d0f901-1
00:54:23.627 --> 00:54:25.520
work with the event records.

963b2f53-8193-4782-8d1b-37b81b890512-0
00:54:26.080 --> 00:54:30.160
OK so I only want to you know
extract the records which are

963b2f53-8193-4782-8d1b-37b81b890512-1
00:54:30.160 --> 00:54:34.172
having an even ID and you know
write them to a database or

963b2f53-8193-4782-8d1b-37b81b890512-2
00:54:34.172 --> 00:54:35.600
write them to a file.

01b9815e-dbb5-4ae3-884a-3eff8a71827e-0
00:54:36.000 --> 00:54:38.539
The rest of the records which
are like odd records, I just

01b9815e-dbb5-4ae3-884a-3eff8a71827e-1
00:54:38.539 --> 00:54:39.400
want to ignore them.

60d5ec2b-d67d-4bb8-a468-e6fa50656e15-0
00:54:39.600 --> 00:54:43.240
OK, so how do we identify the
records?

f01194c2-c799-4361-8061-c513ac8ef38f-0
00:54:43.480 --> 00:54:46.280
How do we transform them and how
to load them?

016d0e90-de2a-4cbb-91d5-d2bdd448e8a4-0
00:54:46.280 --> 00:54:48.680
So that's what you know I want
to talk about.

ae2e0574-74ec-4a18-b661-f84461bcc14b-0
00:54:49.040 --> 00:54:51.000
So I need to go back to studio.

e26db4e5-c722-4646-92bd-855b442dda5f-0
00:54:51.520 --> 00:55:00.830
So in studio let me name the
first batch step as Identify

e26db4e5-c722-4646-92bd-855b442dda5f-1
00:55:00.830 --> 00:55:03.720
even records step.

56637280-e7d7-4433-bc8b-c7303ff32b33-0
00:55:03.960 --> 00:55:06.160
Again we have multiple options
here, I'll cover them.

cbd5901c-2a25-4cbb-b998-a04a5c6d7fdb-0
00:55:06.560 --> 00:55:09.819
So in the first step I want to
add a transform message to

cbd5901c-2a25-4cbb-b998-a04a5c6d7fdb-1
00:55:09.819 --> 00:55:11.280
identify the event record.

961497a3-20b6-4d28-acb1-2424cd1e7715-0
00:55:11.520 --> 00:55:14.200
So I want to use Java only.

0c4ba4bc-b92d-47ed-bd7a-0291be7288a7-0
00:55:14.280 --> 00:55:18.160
So I want to use a simple if and
modulus operation.

b2199ee7-86b8-4352-9ca3-966021308389-0
00:55:18.600 --> 00:55:25.333
So if mod of payload dot ID I
want to divide the ID by two and

b2199ee7-86b8-4352-9ca3-966021308389-1
00:55:25.333 --> 00:55:31.960
if it is reminder is equal to 0
it means it is a even number.

948151ca-939e-43be-a04a-1925b8a43cd3-0
00:55:32.040 --> 00:55:33.920
So I want to say true as an
output.

54fa62c0-9779-4b6a-8e6d-065382a4bb2f-0
00:55:34.360 --> 00:55:39.000
If it is not even it's going to
give me false as a output.

b01583da-f1ce-4849-be42-c3719ca3126f-0
00:55:39.200 --> 00:55:41.440
OK so this is the logic which I
want to use.

af1c6173-b7e0-406e-ba26-ac0e253df470-0
00:55:42.400 --> 00:55:46.439
So what I can do here is so if
this you know receives an ID of,

af1c6173-b7e0-406e-ba26-ac0e253df470-1
00:55:46.439 --> 00:55:49.280
you know thousand thousand is a
even record.

5bbb8fe5-d1b0-424a-8a2b-2594cb7d496d-0
00:55:49.560 --> 00:55:52.200
So the output is going to be
true.

e43904b4-a19e-41fc-9ba0-5659079e1f3f-0
00:55:53.120 --> 00:55:53.280
OK.

0f261247-bd25-4b03-a5b2-cba7d08780df-0
00:55:53.440 --> 00:55:55.280
It's still not running the
preview.

06e9e81e-cfb8-4a37-9d22-9dc6f9cda514-0
00:55:59.160 --> 00:56:01.280
The Java and Jason you have
given right in.

6d385000-6ba3-41bb-aaed-ea52e0bce1ca-0
00:56:01.400 --> 00:56:01.920
No, that's fine.

6612c102-3583-4621-a603-efb65d3afc0a-0
00:56:01.920 --> 00:56:02.200
Java.

2b4d7dca-c0d6-4e91-ab9a-7441088502c2-0
00:56:02.200 --> 00:56:02.840
Java is fine.

c50abaf4-1e11-4818-a2d0-643993960d45-0
00:56:02.960 --> 00:56:05.880
So here we are just setting up
boolean, right.

63c1f9af-eda0-4760-8240-f193aff455b9-0
00:56:05.880 --> 00:56:08.520
So I'm just going to get either

true or false.

6b1b58bf-5f62-4d52-aa15-49bbfb8b94bf-0
00:56:09.200 --> 00:56:11.120
OK, this is taking a lot of
time.

15a7f32e-b2e0-46fe-b960-b0413eef9306-0
00:56:12.960 --> 00:56:13.080
Yeah.

0d47b446-526f-47ca-bc76-2186d846fa68-0
00:56:13.200 --> 00:56:14.840
So you're going to get boolean
as true.

ae856a87-703f-4651-a7fe-637e930ef5d3-0
00:56:15.080 --> 00:56:17.880
So if it is 10,001, of course
it's an odd number.

bb4c0b7c-b5bf-4cda-93ff-b6457532e571-0
00:56:17.880 --> 00:56:19.440
You're going to get output as a
false.

9ab5a0b8-51b4-498a-a6eb-4fedcaaeef41-0
00:56:19.720 --> 00:56:23.063
So I would like to use this
logic to identify the event

9ab5a0b8-51b4-498a-a6eb-4fedcaaeef41-1
00:56:23.063 --> 00:56:26.526
records and the odd records and
I want to store this in a

9ab5a0b8-51b4-498a-a6eb-4fedcaaeef41-2
00:56:26.526 --> 00:56:30.348
variable so you can create a set
variable but again you have to

9ab5a0b8-51b4-498a-a6eb-4fedcaaeef41-3
00:56:30.348 --> 00:56:32.080
drag and drop a set variable.

06f9d12d-e3ff-4762-ae17-d40ea82bdd23-0
00:56:32.280 --> 00:56:35.459
The transform message can be
used for creating a variable as

06f9d12d-e3ff-4762-ae17-d40ea82bdd23-1
00:56:35.459 --> 00:56:35.720
well.

713b79e5-b674-4462-86ac-97f3da1f6918-0
00:56:35.720 --> 00:56:39.240
I'm not sure if you already know
about it so you can click here.

44cb6dd7-81c0-46dd-942c-3a708c8b2b72-0
00:56:39.240 --> 00:56:42.240
Instead of storing it as a
payload you can say.

26bb8b82-14fd-43a5-9d1d-75948ed8dd2d-0

00:56:42.240 --> 00:56:46.600
Plus you can set attributes or
variables also.

f5199010-a021-4ed6-9467-9eb0b6c7aaef-0
00:56:47.000 --> 00:56:54.081
So I just want to say I want to
create a variable of name event

f5199010-a021-4ed6-9467-9eb0b6c7aaef-1
00:56:54.081 --> 00:56:58.840
record so I'll delete the
payload part so.

03203986-c632-4549-94d9-e14232bf1cae-0
00:57:00.000 --> 00:57:04.228
So I have this transform message
where I created a variable so I

03203986-c632-4549-94d9-e14232bf1cae-1
00:57:04.228 --> 00:57:05.920
can use set variable also.

e147cd4d-231c-4531-a5f6-d7c5e2fd6611-0
00:57:05.920 --> 00:57:08.340
But I'm just showing that
transform message can be used

e147cd4d-231c-4531-a5f6-d7c5e2fd6611-1
00:57:08.340 --> 00:57:09.680
for creating variables as well.

82a75ef0-862e-459d-8533-487a5a6b754c-0
00:57:09.960 --> 00:57:13.568
The name of the variable is even
record and it's going to execute

82a75ef0-862e-459d-8533-487a5a6b754c-1
00:57:13.568 --> 00:57:15.920
this logic and store either true
or false.

cb5d58e3-1045-4ba6-90e7-5ff098526987-0
00:57:16.200 --> 00:57:19.400
OK, so let me add a logger.

f2dc2a1d-3a14-41a2-808a-5f00ebe2055d-0
00:57:21.880 --> 00:57:32.960
I'll just log the payload ID and
I want to log the variable as

f2dc2a1d-3a14-41a2-808a-5f00ebe2055d-1
00:57:32.960 --> 00:57:33.840
well.

fd1470a2-dc2e-4f47-8796-df5fe083972e-0
00:57:34.800 --> 00:57:39.160
So let me say wires, dot, even
record.

ed39e309-8704-4480-a499-0ae96e95c2a2-0
00:57:39.560 --> 00:57:44.200
OK, so let's quickly check this
out.

8db9adf2-50f9-468d-b20e-5ad4fc284867-0
00:57:52.400 --> 00:57:54.000
It's getting redeployed.

5648d271-4426-4556-8b72-b815993c383a-0
00:58:12.590 --> 00:58:13.910
Yeah, so it's redeployed.

ffca012c-c343-49c7-aecd-8783a2bf4e8c-0
00:58:13.910 --> 00:58:17.190
So let's make this 10 records.

7f93823d-a0c1-429e-84b6-4300b02a1f4e-0
00:58:17.710 --> 00:58:19.190
OK, I'm sending this 10 records.

4be1cd63-1f94-4065-ba44-fd114d5ce311-0
00:58:20.910 --> 00:58:22.967
It's going to do all the steps
which you have seen earlier,

4be1cd63-1f94-4065-ba44-fd114d5ce311-1
00:58:22.967 --> 00:58:24.750
like creating queues, loading
and all those things.

3f35a1c0-183e-4b3c-beed-699a639468c8-0
00:58:24.750 --> 00:58:25.710
So I won't cover them.

f1415dde-48a4-4808-89a6-017e3af9ca31-0
00:58:25.710 --> 00:58:27.190
So let me clean the console.

7edb3da5-ea45-46cd-b5b4-7e052913a9a3-0
00:58:27.750 --> 00:58:31.485
So now it's going to apply the
logic and it's going to let me

7edb3da5-ea45-46cd-b5b4-7e052913a9a3-1
00:58:31.485 --> 00:58:32.630
just go to the end.

5e5df0fb-d0c3-492c-91a5-08fbc3799e7a-0
00:58:32.630 --> 00:58:35.720
It's going to say one is false,
right?

fa8281a5-350c-44dd-907e-c4c1905a6831-0
00:58:35.720 --> 00:58:36.840
So two is true.

3e097d0f-8623-4fdf-a685-5b9cab2c92b1-0
00:58:36.880 --> 00:58:38.120
I mean two is even, right?

904da599-d8c4-4758-a2ee-ae615493217c-0
00:58:38.120 --> 00:58:40.240
So it's going to just log all
the things.

4d042016-dde4-4d21-8f79-454f1154e6dd-0
00:58:40.240 --> 00:58:40.520
OK.

b8852214-fc98-49e9-ad90-208e969c620b-0
00:58:41.560 --> 00:58:44.680
So now you might ask, so why are
we doing this?

4f3c0f91-6e5b-43b8-9cbf-5d71043661dc-0
00:58:44.920 --> 00:58:48.566
So when I'm doing this, so this
metadata is also going to be

4f3c0f91-6e5b-43b8-9cbf-5d71043661dc-1
00:58:48.566 --> 00:58:50.240
stored in the record itself.

16885a1a-b25b-49ec-a699-b6372f1235e3-0
00:58:50.520 --> 00:58:50.840
OK.

3b569782-2730-4176-97a5-74b3885180e2-0
00:58:51.040 --> 00:58:53.720
So now we completed processing
all the tenant records.

a2ea4d6d-5ef9-4aab-ad5e-d24a4f78ff74-0
00:58:53.960 --> 00:58:57.693
So in this step we are just you
know identifying the event

a2ea4d6d-5ef9-4aab-ad5e-d24a4f78ff74-1
00:58:57.693 --> 00:58:58.200
records.

e3ead3bb-5ee5-4809-bd48-a880326f3fbc-0
00:58:58.320 --> 00:59:02.437
So now the second step is I want
to load this records to an

e3ead3bb-5ee5-4809-bd48-a880326f3fbc-1
00:59:02.437 --> 00:59:05.320
external database or to an
external file.

a9035a93-c091-48b8-aea6-10d25df47f26-0
00:59:05.560 --> 00:59:13.434
So what I can do is I can I can
just drag and drop another batch

a9035a93-c091-48b8-aea6-10d25df47f26-1
00:59:13.434 --> 00:59:14.040
step.

de9b3de9-705c-4eed-a6fe-9f4f086b4705-0
00:59:15.480 --> 00:59:18.080
You can do it in the same step
but it's not advisable.

6aec3578-16b0-48d5-ab95-373440d722f7-0
00:59:18.080 --> 00:59:19.560
So you can use another batch
step.

b162b4d3-7936-447c-93f5-3c55580b548c-0
00:59:19.920 --> 00:59:27.099
Let me name this as Loading data
step and you need to have some

b162b4d3-7936-447c-93f5-3c55580b548c-1
00:59:27.099 --> 00:59:29.680
transform message here.

07f3956a-3bec-4803-abf9-41a69e02d5c0-0
00:59:29.920 --> 00:59:31.920
If you want you can do some
transformation.

b02ddaf8-97fb-4293-ab55-c2ea9a7d1451-0
00:59:32.360 --> 00:59:37.992
OK so I think I'll just
transform the data so I'll just

b02ddaf8-97fb-4293-ab55-c2ea9a7d1451-1
00:59:37.992 --> 00:59:39.200
say name is.

65ebf38e-fc48-4663-a622-5076b3c61d9b-0
00:59:39.440 --> 00:59:45.238
I just want to use uppercase for
the name just for demo purpose

65ebf38e-fc48-4663-a622-5076b3c61d9b-1
00:59:45.238 --> 00:59:50.583
and and just log the ID as well
payload dot ID and ideally

65ebf38e-fc48-4663-a622-5076b3c61d9b-2
00:59:50.583 --> 00:59:53.120
speaking the next connector.

c9a33b40-cbc6-4e03-b234-f23cc5e1cfde-0
00:59:53.120 --> 00:59:57.280
It could be a database insert or
a you know FTP write operation.

f21291de-ce9b-4ccb-b35e-efebe5ef1202-0
00:59:58.200 --> 01:00:01.566
For this example I'm just going
to use a logger to show you how

f21291de-ce9b-4ccb-b35e-efebe5ef1202-1
01:00:01.566 --> 01:00:02.040
it works.

af49e4ad-0df9-4776-86dc-610babe8c498-0
01:00:02.560 --> 01:00:04.920
So let me just log the payload.

b94fa865-493d-45b2-bffb-a580347b7dfc-0
01:00:05.000 --> 01:00:08.040
OK so so yeah I can save this.

7641e1ab-6dfb-4e85-be63-5c30423a30de-0
01:00:08.280 --> 01:00:12.850
But The thing is in the previous
step we identified the event

7641e1ab-6dfb-4e85-be63-5c30423a30de-1
01:00:12.850 --> 01:00:13.440
records.

2b534789-fc0a-43dc-9da7-4d69cb9f535b-0

01:00:13.640 --> 01:00:17.200
In this step I only want to load
the event record.

36b0964b-1743-4fc1-949f-29674c8570ee-0
01:00:17.200 --> 01:00:18.560
So let me say loading.

88e8a8f6-3624-4cbb-b30d-01bbb6892c2a-0
01:00:19.480 --> 01:00:23.405
Event data OK only the event
records I want to load to the

88e8a8f6-3624-4cbb-b30d-01bbb6892c2a-1
01:00:23.405 --> 01:00:26.200
database or you know insert it
to a file.

0a9eadd5-d0bd-426d-bd06-19dade68a431-0
01:00:26.600 --> 01:00:30.280
So how does this step only
process the event records?

ee8c773a-ecfe-491b-b274-09b81094507c-0
01:00:30.600 --> 01:00:34.428
So if you click on the step name
it's going to give you 2 options

ee8c773a-ecfe-491b-b274-09b81094507c-1
01:00:34.428 --> 01:00:36.400
Accept expression, Accept
policy.

f17733f9-6403-4381-ad76-ac6b839b0374-0
01:00:36.760 --> 01:00:38.560
So first talk about accept
expression.

a0bf13f1-a4ff-45b5-8d20-e19603a156c8-0
01:00:38.800 --> 01:00:42.884
So this batch step it's going to
be executed if the accept

a0bf13f1-a4ff-45b5-8d20-e19603a156c8-1
01:00:42.884 --> 01:00:44.200
expression is true.

a5ebf7ad-61b8-4424-839b-4889c7298b7b-0
01:00:44.200 --> 01:00:48.127
So by default you know it's
going to process all the

a5ebf7ad-61b8-4424-839b-4889c7298b7b-1
01:00:48.127 --> 01:00:48.720
records.

9f167aa5-ceaa-4efe-9823-37522e850ddb-0
01:00:49.280 --> 01:00:52.640
You can also define which
records to be processed using

9f167aa5-ceaa-4efe-9823-37522e850ddb-1
01:00:52.640 --> 01:00:53.600
the true option.

4e18eee8-e381-4ac0-b5c1-1c4831f6fd72-0
01:00:53.720 --> 01:00:57.098
So if this evaluates to true,
it's only going to you know,

4e18eee8-e381-4ac0-b5c1-1c4831f6fd72-1
01:00:57.098 --> 01:00:59.160
process the records which are
true.

14351ee9-053a-4ebd-bfcb-23b704eab548-0
01:00:59.480 --> 01:01:02.611
So what I can do is since I'm
setting up a variable in the

14351ee9-053a-4ebd-bfcb-23b704eab548-1
01:01:02.611 --> 01:01:05.000
previous step, I can use that
variable here.

f7ffb12a-32f9-4784-a377-ff7345c140a7-0
01:01:05.000 --> 01:01:07.160
So whereas dot event record.

6d47a71f-1a7b-4f26-857b-a6df69c8d309-0
01:01:07.440 --> 01:01:10.484
So this variable will be
available for all the event

6d47a71f-1a7b-4f26-857b-a6df69c8d309-1
01:01:10.484 --> 01:01:14.161
records which were processed in
the previous step and this will

6d47a71f-1a7b-4f26-857b-a6df69c8d309-2
01:01:14.161 --> 01:01:16.000
be part of their metadata right?

c342018c-1db9-4da3-8d2c-b8909bb996a8-0
01:01:16.000 --> 01:01:21.880
So let me save this, I'll
explain how it works.

c30fe27d-46f3-4517-af35-64847c9268f6-0
01:01:21.880 --> 01:01:25.946
So let me remove them because it
takes a lot of time looping all

c30fe27d-46f3-4517-af35-64847c9268f6-1
01:01:25.946 --> 01:01:26.760
these things.

6b78cfd5-8e08-4459-bf8b-9b52d4621a65-0
01:01:27.240 --> 01:01:29.120
So let me explain what is
happening here.

44f18b2d-8583-4126-91a5-a3b467d0a0ab-0
01:01:29.760 --> 01:01:32.793
You may get a question right how
does this event record

44f18b2d-8583-4126-91a5-a3b467d0a0ab-1
01:01:32.793 --> 01:01:34.960
propagates from this step to

this step?

be8e0c9d-39ca-4584-9514-a749cd88ed9f-0
01:01:35.360 --> 01:01:40.112
So if you remember the slide, if
you remember this, each record

be8e0c9d-39ca-4584-9514-a749cd88ed9f-1
01:01:40.112 --> 01:01:43.677
also has some metadata
associated with it which

be8e0c9d-39ca-4584-9514-a749cd88ed9f-2
01:01:43.677 --> 01:01:48.058
contains one of thing is which
steps are processed and the

be8e0c9d-39ca-4584-9514-a749cd88ed9f-3
01:01:48.058 --> 01:01:52.514
second one is it can also hold
the variables which were you

be8e0c9d-39ca-4584-9514-a749cd88ed9f-4
01:01:52.514 --> 01:01:53.480
know created.

00013973-94c0-413a-bd6a-6e0601f82d5b-0
01:01:53.960 --> 01:01:56.440
You know when it processed in a
batch step right?

a67f0f1d-f8c8-4131-8692-546f54564e8a-0
01:01:56.440 --> 01:01:58.120
So that metadata will be
available.

9385b945-760a-4758-a714-f07340f384e3-0
01:01:58.280 --> 01:02:01.647
So when the record goes to the
second batch step it is we can

9385b945-760a-4758-a714-f07340f384e3-1
01:02:01.647 --> 01:02:04.905
use that metadata and you can
you know use them in your own

9385b945-760a-4758-a714-f07340f384e3-2
01:02:04.905 --> 01:02:05.720
business logic.

061416e3-b25c-44d3-84c8-3fc396d9d2d4-0
01:02:06.040 --> 01:02:09.792
So in this case, I'm just using
that variable in the accept

061416e3-b25c-44d3-84c8-3fc396d9d2d4-1
01:02:09.792 --> 01:02:10.480
expression.

fee08007-7734-4496-9639-3421e9d14694-0
01:02:10.680 --> 01:02:14.258
If the variable is true for that
record, that record will be

fee08007-7734-4496-9639-3421e9d14694-1

01:02:14.258 --> 01:02:15.960
processed in this patch step.

1d88e0f6-cf23-4e44-be75-3626a1406eed-0
01:02:16.360 --> 01:02:18.280
So let's see that in the demo
part.

c25947ed-30e4-4797-8f6a-8a20b1eba9e2-0
01:02:18.520 --> 01:02:20.760
So it is redeployed.

1929e799-0379-4844-adf9-e31775cff63b-0
01:02:21.480 --> 01:02:25.557
So when I make a call, I'm
sending the same 10 calls, so

1929e799-0379-4844-adf9-e31775cff63b-1
01:02:25.557 --> 01:02:27.560
the first step is completed.

e1df82bc-99cf-41b1-ad7e-05a53eabc3d9-0
01:02:27.560 --> 01:02:30.760
Since I didn't add debug points,
I didn't show anything.

58102d88-fbbe-496d-92e2-87e409175f3e-0
01:02:30.760 --> 01:02:32.480
But you can see that true,
false, true, false.

748bfc61-e639-422b-bcc4-1e4d149e4537-0
01:02:32.480 --> 01:02:33.240
It's completed.

3eacc457-83e1-489f-99a1-95c05665845f-0
01:02:33.480 --> 01:02:34.920
So let me clean the console.

fb57558d-82e7-4f39-92de-abf2427bf461-0
01:02:36.280 --> 01:02:38.640
So now what happens in the
second step?

4a259009-3e05-4a76-b401-a5d49c7171be-0
01:02:38.640 --> 01:02:40.240
It should only process the event
record.

5ccec63a-f061-4b22-ba40-8d6cccf3ba2a-0
01:02:40.240 --> 01:02:44.872
So if I do next, let's say the
log, so it's logging the second

5ccec63a-f061-4b22-ba40-8d6cccf3ba2a-1
01:02:44.872 --> 01:02:47.960
record ID 2, then it's going to
log ID 4.

dca763ef-9290-47f5-87e6-f021a83227a0-0
01:02:47.960 --> 01:02:50.480
So these are all event records,
even ID records.

17d7197e-d1fd-4e61-ba91-fe53b91c4e3f-0

01:02:50.640 --> 01:02:55.680
So if I do 6-8 and ten, that's
it and the processing stopped.

557e270e-cc92-4bf7-878e-2603d9bd4365-0
01:02:55.880 --> 01:02:59.384
So it's only going to, you know,
process the event records in the

557e270e-cc92-4bf7-878e-2603d9bd4365-1
01:02:59.384 --> 01:03:02.888
second batch step because we are
using an accept expression which

557e270e-cc92-4bf7-878e-2603d9bd4365-2
01:03:02.888 --> 01:03:04.800
was set in the previous batch
step.

508d8adc-a6ad-4db9-b379-6dcfcb3b037f-0
01:03:05.520 --> 01:03:05.720
OK.

392ea92c-9f03-4060-ad90-33586c7ce9e1-0
01:03:05.720 --> 01:03:10.028
So that's how you can, you know,
identify fewer records based on

392ea92c-9f03-4060-ad90-33586c7ce9e1-1
01:03:10.028 --> 01:03:14.337
some business logic and you can
work on those individual records

392ea92c-9f03-4060-ad90-33586c7ce9e1-2
01:03:14.337 --> 01:03:18.049
in the second batch tab and
insert them or you know, do

392ea92c-9f03-4060-ad90-33586c7ce9e1-3
01:03:18.049 --> 01:03:19.639
whatever you want to do.

517d3b2e-173d-42f2-bcb9-c603341cb720-0
01:03:20.920 --> 01:03:21.680
Any questions?

1db0641b-454f-4a53-a0d2-3ab7a6096f19-0
01:03:21.840 --> 01:03:28.040
Sofa no, send that.

5725331b-5d37-4597-b31a-4c6fadfe7e2f-0
01:03:30.560 --> 01:03:30.960
OK.

0c87545e-c913-433d-9f39-9fdb7fbecd90-0
01:03:31.720 --> 01:03:33.720
So again, there is a no, I'm
sorry, go ahead.

459e747c-c91c-466f-b071-f9cbf8f6edf3-0
01:03:34.840 --> 01:03:38.755
2nd batch is going to utilize
the the queues which are already

459e747c-c91c-466f-b071-f9cbf8f6edf3-1

01:03:38.755 --> 01:03:42.360
created in the batch one or yes,
no, no, the same queues.

351666ad-f4b9-4151-91b2-b671efab0040-0
01:03:42.840 --> 01:03:45.900
So the batch step whenever,
yeah, the 2nd batch step you

351666ad-f4b9-4151-91b2-b671efab0040-1
01:03:45.900 --> 01:03:46.920
mean to say, right.

2023e0ab-582f-4315-a173-3fb419a5b724-0
01:03:47.120 --> 01:03:50.228
So yeah, when the record
completes processing in batch

2023e0ab-582f-4315-a173-3fb419a5b724-1
01:03:50.228 --> 01:03:53.957
step one, it will be pushed back
to the same queue so that it can

2023e0ab-582f-4315-a173-3fb419a5b724-2
01:03:53.957 --> 01:03:56.839
be fetched and processed in the
next set of steps.

3bd1a96a-d3b1-4636-bc11-e5d355198d11-0
01:03:57.520 --> 01:03:57.920
OK.

02583f26-e9bb-4b4f-a402-7385d56ff585-0
01:03:57.920 --> 01:03:59.240
OK, Yeah.

f6ba90ee-9bdc-4328-a503-c81d47cdcb12-0
01:04:00.120 --> 01:04:01.880
So I think you missed the
animation.

00d56d7e-3a1b-4c21-b42e-d5a54b08d768-0
01:04:01.880 --> 01:04:03.120
So I think it does small
animation.

ef1e0da8-849e-4f61-96ae-69c076fcc4c0-0
01:04:03.120 --> 01:04:06.960
So yeah, where is this?

62f8edeb-a59f-4412-bb2d-dd8f12c21a3c-0
01:04:08.240 --> 01:04:10.920
OK, so now there is a small
issue over here.

d1646637-23da-4ab4-925b-b6646733d0de-0
01:04:10.920 --> 01:04:14.910
So if you remember this logger,
it was logging each record

d1646637-23da-4ab4-925b-b6646733d0de-1
01:04:14.910 --> 01:04:17.480
individually instead of five
records.

367960a9-3fce-4013-bac5-d9e6a6383930-0
01:04:17.560 --> 01:04:20.820
So here we have only five event
records, so the logger was

367960a9-3fce-4013-bac5-d9e6a6383930-1
01:04:20.820 --> 01:04:21.760
executed 5 times.

bbf6a99c-1c8b-4533-84e9-5c5af2fc4269-0
01:04:22.120 --> 01:04:25.707
Let's assume you're working with
a million records, out of which

bbf6a99c-1c8b-4533-84e9-5c5af2fc4269-1
01:04:25.707 --> 01:04:28.080
you might be having 500,000
event records.

39ee47c5-a4ed-40ba-9a63-37b01ab39cdc-0
01:04:28.520 --> 01:04:32.846
And if you're using a database
insert here for those 500,000

39ee47c5-a4ed-40ba-9a63-37b01ab39cdc-1
01:04:32.846 --> 01:04:37.314
records, it has to open 500,000
connections and insert 500,000

39ee47c5-a4ed-40ba-9a63-37b01ab39cdc-2
01:04:37.314 --> 01:04:38.520
times separately.

c50126e8-31eb-41fc-8513-d399beb79a8f-0
01:04:38.800 --> 01:04:42.447
Just imagine what would be the
load on the external database,

c50126e8-31eb-41fc-8513-d399beb79a8f-1
01:04:42.447 --> 01:04:42.800
right.

aa72c0a8-df71-42f0-9129-e4ed56c21a6b-0
01:04:42.800 --> 01:04:44.240
So it's going to crash at one
point.

5a4b7234-5fd2-4c66-9289-079720bbf1aa-0
01:04:44.480 --> 01:04:46.080
So how do we, how do we avoid
that?

dacb0f8f-9cde-44c0-8e1f-4e43f385ad2f-0
01:04:46.280 --> 01:04:47.200
So that is the place.

1ad76797-87e6-45ce-bace-d4355a4798e1-0
01:04:47.200 --> 01:04:49.866
I mean this is the place where
you can make use of the

1ad76797-87e6-45ce-bace-d4355a4798e1-1
01:04:49.866 --> 01:04:50.400
aggregator.

32f77453-cc64-4fd6-8b9a-412a6a2d307c-0
01:04:50.440 --> 01:04:52.120
So what is this aggregator?

88acdc30-8239-4e11-b314-4f303bdc45cc-0
01:04:52.600 --> 01:04:55.852
So aggregator is a concept which
can aggregate N number of

88acdc30-8239-4e11-b314-4f303bdc45cc-1
01:04:55.852 --> 01:04:58.720
records and process them you
know in a single call.

bfb85e82-3a53-462a-be78-f931bf8e4312-0
01:04:59.160 --> 01:05:03.560
So for aggregator you have to
drag and drop this aggregator

bfb85e82-3a53-462a-be78-f931bf8e4312-1
01:05:03.560 --> 01:05:04.000
scope.

1627d828-e9a5-4e00-97b1-3c4703986b8c-0
01:05:04.000 --> 01:05:08.200
Here you have two options
streaming and the size.

64505bd1-531e-409d-8aeb-f7ca3e27ba29-0
01:05:08.480 --> 01:05:12.559
So if I give size is equal to
100, it's going to load 100

64505bd1-531e-409d-8aeb-f7ca3e27ba29-1
01:05:12.559 --> 01:05:16.779
records and log all the 100
records once Or if you're using

64505bd1-531e-409d-8aeb-f7ca3e27ba29-2
01:05:16.779 --> 01:05:21.351
a database insert it's going to
insert all the 100 records using

64505bd1-531e-409d-8aeb-f7ca3e27ba29-3
01:05:21.351 --> 01:05:23.040
a single insert command.

55631546-cbfb-4579-962b-c1568e48d0a2-0
01:05:23.040 --> 01:05:25.080
So you have this connectors
right?

c68fe0b6-f55d-43b3-809e-d6b1c169aa4a-0
01:05:25.080 --> 01:05:28.386
So if you add the database
connector you have this bulk

c68fe0b6-f55d-43b3-809e-d6b1c169aa4a-1
01:05:28.386 --> 01:05:28.800
insert.

cfdae237-35bf-4347-bc80-0f806ae0e23c-0
01:05:29.120 --> 01:05:32.932
So bulk insert is used in this
batch aggregator which can

cfdae237-35bf-4347-bc80-0f806ae0e23c-1
01:05:32.932 --> 01:05:37.008
insert you know, 100 records
using a single insert command or

cfdae237-35bf-4347-bc80-0f806ae0e23c-2
01:05:37.008 --> 01:05:38.520
1000 or 10,000 records.

11d9fbbb-73a0-41b3-b859-3abcec7ec590-0
01:05:38.640 --> 01:05:41.240
So in this way you can improve
the performance of the database.

7cb5f47d-85b5-491a-b4fe-431a1849ccfb-0
01:05:41.240 --> 01:05:43.400
You don't need to make hundred
separate inserts.

9f55a556-eb0b-4100-953f-de69e4675a52-0
01:05:43.480 --> 01:05:46.560
OK, so that's the use of the
aggregator.

6f10bed7-a04c-46b0-a3d3-7f586f2c82f7-0
01:05:47.120 --> 01:05:48.600
So let me quickly show you this.

d123c3c1-1e43-4e95-8e25-ee439f730663-0
01:05:48.600 --> 01:05:50.720
So I'll also use a streaming
option.

f0c5206f-2686-456a-866f-24604917b3bf-0
01:05:50.720 --> 01:05:54.120
Also let me just make a small
change.

12502191-1f3c-42e1-bd08-b1ec743d43fb-0
01:05:54.880 --> 01:05:56.840
OK, one question say that see
you said it does.

33b69e75-7ca0-448a-8cf8-5f6e148f9159-0
01:05:57.120 --> 01:05:59.200
OK, let's say the aggregator you
said.

31ddfbf2-482d-47e1-8b31-45e5fd7a2883-0
01:05:59.560 --> 01:06:02.200
So it is going to withhold all
the.

2c713095-7ea6-4445-8302-462369b6ab31-0
01:06:03.600 --> 01:06:06.992
Suppose you have 10 records,
it's like the 10 records, 10

2c713095-7ea6-4445-8302-462369b6ab31-1
01:06:06.992 --> 01:06:10.559
records if you want to aggregate
and only 10 records will be

2c713095-7ea6-4445-8302-462369b6ab31-2

01:06:10.559 --> 01:06:11.320
there, right.

b6e790fa-9cfb-43ee-a717-ed9828a172cd-0
01:06:11.320 --> 01:06:15.040
So all the time it will wait a
complete all the last steps and

b6e790fa-9cfb-43ee-a717-ed9828a172cd-1
01:06:15.040 --> 01:06:17.520
it will wait and then I'll, I'll
explain.

9d20f2a2-e311-4a80-b91f-2c58b4991a91-0
01:06:18.520 --> 01:06:21.840
So this transfer message, I mean
it is going to wait here itself.

c3906b5f-fb68-46e1-8655-85842f35819f-0
01:06:21.840 --> 01:06:25.040
So in our case I think, oh, OK,
I'm using hundred.

215d1818-5c84-4fa3-a682-32089e461c93-0
01:06:25.080 --> 01:06:27.920
I don't want to use hundred, I
want to use three.

51c83804-39b4-46b6-bdd6-14f2c4b6dadc-0
01:06:28.080 --> 01:06:29.600
I just want to show you how it
works.

846b2de8-dcec-4e88-b865-31cb2237c0f6-0
01:06:29.960 --> 01:06:31.080
Let me save this again.

bf05b36a-5d2b-48e6-9d5f-36edea8af1da-0
01:06:33.040 --> 01:06:33.680
Is it saved?

4ef5896c-eabc-4351-b021-6f14eef7e001-0
01:06:33.680 --> 01:06:33.880
No.

093de7d9-3804-4fca-a8e0-2cab4df0ae95-0
01:06:33.880 --> 01:06:36.960
Basically it's going to 333
batches of three aggregates.

a24c5163-60fe-436a-9ebe-273c8a5b5153-0
01:06:36.960 --> 01:06:37.680
Yes, 3 batches.

847063b2-f8f8-48d6-b45e-976b7dfcc915-0
01:06:37.680 --> 01:06:39.280
Yeah, it's going to 11 extract.

26254b5e-5d39-4746-b47b-b6da021f673f-0
01:06:39.320 --> 01:06:42.200
OK yeah, four before output.

7712c4bd-bba6-43cd-a96e-002c5e6b0ea8-0
01:06:43.000 --> 01:06:44.840
So let me let me show you how it

works.

cf268302-8655-4f68-98b6-1e8a27633c37-0
01:06:45.720 --> 01:06:47.802
So first it's going to, I mean
here we have a transform

cf268302-8655-4f68-98b6-1e8a27633c37-1
01:06:47.802 --> 01:06:48.360
message, right.

f0d47ae1-97f9-4275-9b0f-a5f8f47014d4-0
01:06:48.360 --> 01:06:51.280
So this transform message, it's
going to process 3 records.

b795b3f6-e7e4-486d-8e80-31550a64e5a0-0
01:06:51.440 --> 01:06:55.188
Once it completes processing 3
records, those 3 records will be

b795b3f6-e7e4-486d-8e80-31550a64e5a0-1
01:06:55.188 --> 01:06:58.820
inserted or logged using the
batch aggregator because we gave

b795b3f6-e7e4-486d-8e80-31550a64e5a0-2
01:06:58.820 --> 01:06:59.640
the size as 3.

624d6647-6cc4-4eeb-9152-99e8afc69726-0
01:06:59.840 --> 01:07:03.013
Then it's going to pick up the
next two records which are

624d6647-6cc4-4eeb-9152-99e8afc69726-1
01:07:03.013 --> 01:07:06.515
available, it processes them and
then those two records will be

624d6647-6cc4-4eeb-9152-99e8afc69726-2
01:07:06.515 --> 01:07:10.072
you know, sent to the aggregator
so that you know it can be, you

624d6647-6cc4-4eeb-9152-99e8afc69726-3
01:07:10.072 --> 01:07:11.440
know, inserted or logged.

07390cef-dbac-4886-932a-ffca7b3c1f22-0
01:07:11.640 --> 01:07:15.428
So the aggregator is going to
hold this many records only when

07390cef-dbac-4886-932a-ffca7b3c1f22-1
01:07:15.428 --> 01:07:18.736
it, you know reaches this
threshold, it's going to you

07390cef-dbac-4886-932a-ffca7b3c1f22-2
01:07:18.736 --> 01:07:20.480
know do the insert or log in.

759c6213-66c7-478b-80ac-bd09999d42ee-0
01:07:20.880 --> 01:07:22.160

So it's going to hold
internally.

7b50fed1-c18c-4128-9c73-f8a6d6b7b613-0
01:07:22.800 --> 01:07:24.240
So let me show you how it works.

17fd0d35-9029-4622-bc34-1a66907cb066-0
01:07:24.240 --> 01:07:25.120
So it's same.

19fdb6b7-c445-4289-b739-24b6e00f27ea-0
01:07:25.120 --> 01:07:27.883
I think you can remove that
expression through otherwise it

19fdb6b7-c445-4289-b739-24b6e00f27ea-1
01:07:27.883 --> 01:07:30.323
you will not get all that
because you put the event,

19fdb6b7-c445-4289-b739-24b6e00f27ea-2
01:07:30.323 --> 01:07:30.600
right?

09099f62-99aa-491a-b29a-9fb66f088b7e-0
01:07:31.920 --> 01:07:32.360
That's fine.

926daef5-4732-4b2f-a93e-cda6dbf08374-0
01:07:32.360 --> 01:07:35.387
I want to show you only with the
event records how it works, I'll

926daef5-4732-4b2f-a93e-cda6dbf08374-1
01:07:35.387 --> 01:07:35.800
show you.

111837a0-8d5b-4cbe-9ba7-ebad4faac552-0
01:07:35.800 --> 01:07:38.351
OK, so even even in event
records we have 5 records,

111837a0-8d5b-4cbe-9ba7-ebad4faac552-1
01:07:38.351 --> 01:07:38.640
right?

7e30d1a1-844d-4c20-9990-113ef08e98d8-0
01:07:38.840 --> 01:07:41.000
So I want to insert them in
batches of three.

f9d9bfe6-8e35-43e1-a3df-7cb4180865fd-0
01:07:41.720 --> 01:07:43.840
OK, not all of them individually
or once.

f446d2b8-bef9-4bc1-8dd4-e143268320e9-0
01:07:43.840 --> 01:07:44.840
So let me make a call.

c67c0ce6-ec18-411f-937a-a6cf32a83aa0-0
01:07:47.760 --> 01:07:50.400
It comes to the yeah second
step.

37f42443-3a59-43a9-875c-4e4d7417e8ce-0
01:07:50.760 --> 01:07:54.691
In the second step, if you see
this transfer message will be

37f42443-3a59-43a9-875c-4e4d7417e8ce-1
01:07:54.691 --> 01:07:58.622
executed three times before the
logger gets executed if I do

37f42443-3a59-43a9-875c-4e4d7417e8ce-2
01:07:58.622 --> 01:07:58.880
123.

a3f93dfc-77ef-48ec-8ae4-fe2a40bffe71-0
01:07:59.000 --> 01:08:03.202
So once it fetches or it
accumulates 3 records, it's

a3f93dfc-77ef-48ec-8ae4-fe2a40bffe71-1
01:08:03.202 --> 01:08:07.960
going to log all the three
records in a single log message.

625086ae-01d6-4a8c-a360-28acb25510c5-0
01:08:08.000 --> 01:08:11.545
If it's a insert, 3 records will
be inserted using the single

625086ae-01d6-4a8c-a360-28acb25510c5-1
01:08:11.545 --> 01:08:15.090
insert bulk insert command and
then it's going to process the

625086ae-01d6-4a8c-a360-28acb25510c5-2
01:08:15.090 --> 01:08:16.120
remaining records.

2b82c6e0-9865-4ef8-8d15-4dcd1dab06fb-0
01:08:16.120 --> 01:08:18.640
So we only have two more records
remaining.

9279ea40-cc50-45bc-a864-25295838a8a8-0
01:08:18.640 --> 01:08:22.560
So those two records will be,
you know, logged.

4fae0d33-cf8b-47e1-9eac-2bed0f869121-0
01:08:23.000 --> 01:08:24.320
So that is how the aggregator
works.

cfed13f9-5d00-49c4-bbb3-0652cf5bba2b-0
01:08:24.320 --> 01:08:27.525
It's going to aggregate the
number of records based on the

cfed13f9-5d00-49c4-bbb3-0652cf5bba2b-1
01:08:27.525 --> 01:08:30.840
number you give and insert them
or uploads them all at once.

454c86bf-ca3c-4ce4-be9f-7ead16d7b805-0

01:08:35.720 --> 01:08:36.280
Makes sense.

efe0010e-b346-4e5f-8e4f-5349316db979-0
01:08:37.120 --> 01:08:41.606
Yes it does see that like first
like at recent when it when it

efe0010e-b346-4e5f-8e4f-5349316db979-1
01:08:41.606 --> 01:08:45.949
started 3 records OK transfer
message and when it was trying

efe0010e-b346-4e5f-8e4f-5349316db979-2
01:08:45.949 --> 01:08:50.080
to like update in the logger OK
during it, it got failed.

ecf41f94-8c5a-47d5-986c-c9ea754f43a6-0
01:08:50.360 --> 01:08:53.760
So you can start from again
second right at that time no no.

58b93021-415d-488f-b4db-0ae031924a5a-0
01:08:53.760 --> 01:08:57.153
If that is failed you have to do
some error handling, so it's

58b93021-415d-488f-b4db-0ae031924a5a-1
01:08:57.153 --> 01:09:00.000
going to start by the next set
of records 2 and 10.

6e946243-e5fa-4454-acbf-97b0e7596201-0
01:09:00.840 --> 01:09:03.641
So if something failed here,
like inserting failed or

6e946243-e5fa-4454-acbf-97b0e7596201-1
01:09:03.641 --> 01:09:07.014
connecting to a database failed,
you have to make sure that that

6e946243-e5fa-4454-acbf-97b0e7596201-2
01:09:07.014 --> 01:09:08.000
record is not lost.

7c1fc41c-f473-4b66-8fec-e9e4dddb0048-0
01:09:08.000 --> 01:09:11.286
So you might use you you should
be using some, you know queues

7c1fc41c-f473-4b66-8fec-e9e4dddb0048-1
01:09:11.286 --> 01:09:14.728
so that you know you can send it
to some dead letter queue or you

7c1fc41c-f473-4b66-8fec-e9e4dddb0048-2
01:09:14.728 --> 01:09:17.962
have to insert it to a you know
what we say, mock database so

7c1fc41c-f473-4b66-8fec-e9e4dddb0048-3
01:09:17.962 --> 01:09:21.301
that you know you can reevaluate

why it failed so that you have

7c1fc41c-f473-4b66-8fec-e9e4dddb0048-4
01:09:21.301 --> 01:09:22.240
to do it manually.

0d7da5ac-2202-4fd1-8206-8364ddc9a541-0
01:09:22.240 --> 01:09:25.603
So the transfer message will
continue processing the next set

0d7da5ac-2202-4fd1-8206-8364ddc9a541-1
01:09:25.603 --> 01:09:26.200
of records.

86e68d5b-7abd-424d-91be-67ca6c05b750-0
01:09:27.080 --> 01:09:28.600
It goes like it on complete
right?

7315ccbc-bae8-4be4-9134-8b6122d26163-0
01:09:28.640 --> 01:09:30.200
Until on it completely.

c86450db-7ad6-45dd-b13c-e3e2a6441b5d-0
01:09:30.200 --> 01:09:32.800
That next phase the data will be
available on the queue, right?

b4b94dfd-c247-4b2f-be0d-8997689273ef-0
01:09:32.800 --> 01:09:33.840
And persistent queue, right?

4fd2d085-61f0-44da-911e-2fae07a96e86-0
01:09:35.160 --> 01:09:35.480
Yes.

6f464a1f-729a-4eec-891c-eac966b48812-0
01:09:35.480 --> 01:09:37.120
But here the record is failing,
right?

a8aad901-dbb5-4ac2-b24a-c63d4bdeb980-0
01:09:37.120 --> 01:09:40.163
So after the record is failing,
if it goes outside it will be

a8aad901-dbb5-4ac2-b24a-c63d4bdeb980-1
01:09:40.163 --> 01:09:41.440
removed from the VM queue.

dc2032ab-9837-47a2-8161-4b22c77d3d0c-0
01:09:41.440 --> 01:09:42.880
So you don't want to lose the
data.

84a3350b-26e2-4aaf-a80f-491d4b9f914d-0
01:09:42.880 --> 01:09:46.290
So you have to use some Tri
scope and use some error

84a3350b-26e2-4aaf-a80f-491d4b9f914d-1
01:09:46.290 --> 01:09:48.800
handling to persist it somewhere
else.

69c0e496-f358-480b-a449-007bdbd35579-0
01:09:49.280 --> 01:09:50.120
Yeah, time again.

bc21c923-cd15-4d81-94fa-190130e58aa0-0
01:09:50.320 --> 01:09:50.640
It's it.

4e0c6415-aa5c-47ef-95fd-fcff0d83889d-0
01:09:50.680 --> 01:09:52.560
It comes under error handling in
that case.

cab4734c-902f-4fe6-8069-9823536cebae-0
01:09:52.560 --> 01:09:53.920
OK, OK, OK, yeah.

da4cee29-0248-4c10-aa58-46732d9be7e2-0
01:09:53.920 --> 01:09:56.781
Because if you do, if you don't
do error handling here it comes

da4cee29-0248-4c10-aa58-46732d9be7e2-1
01:09:56.781 --> 01:09:56.960
out.

e7c6fb3b-8ad8-4d9d-bee4-cbc4b5a35ff2-0
01:09:56.960 --> 01:09:59.377
Since there are no more steps,
it will be deleted from the

e7c6fb3b-8ad8-4d9d-bee4-cbc4b5a35ff2-1
01:09:59.377 --> 01:10:00.320
persistent VMQ as well.

1b2b1695-b343-4419-aab1-341f21c62572-0
01:10:00.320 --> 01:10:01.240
Yeah, VMQ, you're correct.

8af048e7-d22a-4808-bc1b-71ede64e90fd-0
01:10:01.280 --> 01:10:05.120
Yeah, OK, right.

a4ecc729-c2ed-4528-a3a1-4818c835be7a-0
01:10:05.120 --> 01:10:06.840
So that's about aggregator.

655d3ef2-6fd2-44e3-8cd6-5cee0c84f963-0
01:10:06.960 --> 01:10:09.902
So in aggregator, I think I also
wanted to talk about the

655d3ef2-6fd2-44e3-8cd6-5cee0c84f963-1
01:10:09.902 --> 01:10:12.844
streaming option as well, like
what is this and why it is

655d3ef2-6fd2-44e3-8cd6-5cee0c84f963-2
01:10:12.844 --> 01:10:13.200
useful.

6789bdcb-5ec4-4795-a88a-c56df57a9ede-0
01:10:13.640 --> 01:10:18.654
So first let me explain what is

the downside of using aggregator

6789bdcb-5ec4-4795-a88a-c56df57a9ede-1
01:10:18.654 --> 01:10:19.040
size.

75c72554-de95-4b6d-a906-2efe3eaf3a7b-0
01:10:19.240 --> 01:10:22.600
So this aggregator size is
useful for databases.

b09f340f-9b64-45b1-a6ea-658569f2eab1-0
01:10:22.600 --> 01:10:26.903
OK, if you use 300 it's going to
you know insert 300 records at

b09f340f-9b64-45b1-a6ea-658569f2eab1-1
01:10:26.903 --> 01:10:27.240
once.

59894467-3b02-457d-bce6-19444126ccae-0
01:10:27.440 --> 01:10:31.520
But it might not work for FTP or
SFTP or file based services.

c0874946-5470-4f3a-8088-1c1417e48d21-0
01:10:31.760 --> 01:10:36.280
The reason is let me use you
know write connector.

be52ccbb-dded-4789-8baa-ee0f50edf42e-0
01:10:36.280 --> 01:10:40.667
I want to write the data so I
want to write the data to source

be52ccbb-dded-4789-8baa-ee0f50edf42e-1
01:10:40.667 --> 01:10:42.200
main resources folder.

cb9b5d42-95a2-41fb-861a-32c0a5d4c64f-0
01:10:42.200 --> 01:10:43.480
Let me copy the path.

7e07cca8-a4b3-41b4-97d3-1c409b15aa7a-0
01:10:47.480 --> 01:10:50.514
So here I just want to create
something called as test dot

7e07cca8-a4b3-41b4-97d3-1c409b15aa7a-1
01:10:50.514 --> 01:10:51.080
Jason file.

dfed764c-aae1-4adc-b870-ab6ce61f48da-0
01:10:51.640 --> 01:10:56.680
I want to convert the data to
Jason and save it in a file.

b9995503-413c-4147-89c8-29d7e55c6bb7-0
01:10:56.800 --> 01:10:59.560
OK application dot Jason
payload.

a5023fed-2f7d-4d23-a60f-fedf251a1e01-0
01:10:59.880 --> 01:11:04.010
OK and if the you know directory

does not exist, it's going to

a5023fed-2f7d-4d23-a60f-fedf251a1e01-1
01:11:04.010 --> 01:11:06.240
create the directory on the
file.

1f80d576-4acb-4754-ac34-2e0b1c0816b8-0
01:11:06.520 --> 01:11:09.736
And I want to do something
called as append because we will

1f80d576-4acb-4754-ac34-2e0b1c0816b8-1
01:11:09.736 --> 01:11:11.720
be appending multiple records
right?

e9a4dcd2-6372-4335-910b-db7012dee242-0
01:11:11.840 --> 01:11:13.680
So I want to keep on appending
the records.

af0ced35-4aa6-461e-9149-ee793905d710-0
01:11:13.920 --> 01:11:19.226
So let's see what is the, you
know, disadvantage of using

af0ced35-4aa6-461e-9149-ee793905d710-1
01:11:19.226 --> 01:11:23.160
aggregator size for file based
connectors.

66d75183-41f6-4c96-bc09-952a201e7ada-0
01:11:23.480 --> 01:11:27.330
Aggregator size is perfect for
database inserts but not ideal

66d75183-41f6-4c96-bc09-952a201e7ada-1
01:11:27.330 --> 01:11:28.200
for the files.

95cc15ae-1bc9-4948-ac19-51c999389039-0
01:11:28.280 --> 01:11:33.160
So let's see why if it's taking
a time.

12b0d302-aa53-4a88-903d-b30ba61dca8d-0
01:11:36.440 --> 01:11:39.702
I think after this I only have
one more concept left like what

12b0d302-aa53-4a88-903d-b30ba61dca8d-1
01:11:39.702 --> 01:11:43.120
do we do if there is some error,
so I'll talk about that as well.

c100b4bf-ef97-431b-9b4c-7302ada6a731-0
01:11:51.120 --> 01:11:51.800
Is it deployed?

a852082b-eb03-481a-ac13-777620bfcb38-0
01:11:51.800 --> 01:11:52.800
No, not yet deployed.

4a87b554-a4db-4a3c-9bec-9d523bb33133-0

01:11:53.440 --> 01:11:57.000
Oh yeah, so let me see the PPT.

78645eb6-2c84-430e-bd64-fa7b42e401da-0
01:11:58.280 --> 01:12:03.440
So OK, I think or I think
variables also is there.

95922180-402a-4328-8d16-4016e0fa4515-0
01:12:04.840 --> 01:12:06.080
So let me clean the console.

822ea939-4123-474e-970a-6a8b6be01300-0
01:12:06.920 --> 01:12:11.000
Now if I go back, so I'll make
the same 10 calls.

199a7a58-75c9-4d5a-8e53-73ffd250f637-0
01:12:13.520 --> 01:12:16.560
So it's going to do transform
SS123.

cf1351c4-eeda-4b69-a972-3c5b45229263-0
01:12:16.600 --> 01:12:17.880
It comes to the file, right?

75588fa0-a917-438a-a9c6-5430a80e8f02-0
01:12:19.000 --> 01:12:20.040
It's going to write the file.

d8e2b402-f0b4-42b7-a912-2c9a97c99ab5-0
01:12:20.040 --> 01:12:21.600
So let me show you the file as
well.

f2cfcecb-a9df-43e4-b38c-c7522ecbec36-0
01:12:21.640 --> 01:12:25.886
So if I go to my source main
resources, refresh, a new Jason

f2cfcecb-a9df-43e4-b38c-c7522ecbec36-1
01:12:25.886 --> 01:12:27.000
file is created.

59a66d94-0dd7-4c5d-89df-3b0911d8b073-0
01:12:27.440 --> 01:12:31.000
Let me open this and it should
contain the Jason.

e3bd150f-7887-489b-a30c-c71cf5c5a23b-0
01:12:31.120 --> 01:12:32.520
OK 246.

91eac8ad-2c3b-469a-8712-61e4a2b3ec51-0
01:12:32.800 --> 01:12:34.840
So now I added append.

e0cb69ad-b9eb-43cd-88ea-11c3e8a84e58-0
01:12:34.880 --> 01:12:39.040
So let's see what happens if I
go back to the debug mode.

c5f93d3f-3e55-40b5-bd72-3dce0bb892b2-0
01:12:39.040 --> 01:12:43.268
And if I consume the next two

records, it's going to write the

c5f93d3f-3e55-40b5-bd72-3dce0bb892b2-1
01:12:43.268 --> 01:12:45.080
file OK, there is no error.

2ac9aa56-a4ab-49d5-9bb4-3d46a4b1abba-0
01:12:45.360 --> 01:12:49.200
If I go to test Jason let me use
this.

635a7018-89ed-4d10-a314-e48b293d62b7-0
01:12:49.400 --> 01:12:52.240
So two more records are
appended.

01a0d64e-9795-4d95-bfde-46549d7ed006-0
01:12:52.480 --> 01:12:54.760
But is this a valid Jason?

3e55a072-25ce-41bd-98be-fc86b83ded2f-0
01:12:55.360 --> 01:12:57.080
It's not a valid Jason right?

94867841-9b42-4fa9-9ec0-a07d3aaaf7fe-0
01:12:57.360 --> 01:13:01.069
So that is the you know downside
of using aggregator size for

94867841-9b42-4fa9-9ec0-a07d3aaaf7fe-1
01:13:01.069 --> 01:13:04.600
file based operations because
it's simply going to append.

813bc74f-ad0d-4c58-b22a-2790ab9ecf4c-0
01:13:04.600 --> 01:13:07.240
It does not know that you know
it's a valid Jason right?

2f564fb3-5e11-42ea-a530-057b07daac5d-0
01:13:07.480 --> 01:13:09.280
So that is the reason for file
based.

2d745f3b-061d-49e3-89ad-ed0bc3bb7176-0
01:13:09.280 --> 01:13:10.680
So let me just clean this.

20a6bc05-d361-4064-9399-ebbce2fc380c-0
01:13:11.280 --> 01:13:15.944
OK, I want to redo this so you
can make use of the streaming

20a6bc05-d361-4064-9399-ebbce2fc380c-1
01:13:15.944 --> 01:13:16.480
option.

a1445d63-bde3-439a-be8b-3133ef2f6050-0
01:13:16.760 --> 01:13:20.120
Again, you can either use
aggregator size or streaming.

95fa3eb1-08e7-4a34-9a30-10805f9adfbc-0
01:13:20.120 --> 01:13:21.480
You can't use both of them.

cfde7b37-4b46-4eaf-901e-4d4863743c73-0
01:13:21.760 --> 01:13:24.889
So I can I should be removing
the size I need to use

cfde7b37-4b46-4eaf-901e-4d4863743c73-1
01:13:24.889 --> 01:13:25.480
streaming.

ac233c50-0047-4ce6-92eb-48d2497232e7-0
01:13:26.720 --> 01:13:29.740
So what happens in streaming is
as soon as it receives the

ac233c50-0047-4ce6-92eb-48d2497232e7-1
01:13:29.740 --> 01:13:31.840
record it's going to be you know
insert.

3e3e257f-7b73-4970-b37d-dd7a279e6e5f-0
01:13:31.920 --> 01:13:33.240
I mean it's going to be returned
to the file.

dfe708d1-e3c2-4166-9c25-77a727a2e68b-0
01:13:33.920 --> 01:13:38.640
So let's see in that case the
why do you want an aggregator.

74481f92-5f13-4766-9bba-d33310f041d2-0
01:13:38.680 --> 01:13:41.179
There you can actually put next
to the activity in the

74481f92-5f13-4766-9bba-d33310f041d2-1
01:13:41.179 --> 01:13:43.360
transformation button app and
write write file.

ffba32dd-2c37-47eb-8ed5-88c5d011014e-0
01:13:44.680 --> 01:13:45.880
Yes, even that can be done.

04f81435-630f-4647-b1c7-2fd7ddbb4999-0
01:13:45.880 --> 01:13:46.280
Yes, yes.

93ea14eb-efae-49a6-a4e7-3d0e814a47d7-0
01:13:46.480 --> 01:13:50.384
But streaming I guess it's a bit
faster compared to the OK, the

93ea14eb-efae-49a6-a4e7-3d0e814a47d7-1
01:13:50.384 --> 01:13:51.360
the regular one.

409adf66-571f-4595-a68a-d169543493e9-0
01:13:51.400 --> 01:13:55.320
So I don't exactly remember the
correct point.

5fcea510-7d01-437c-b67e-2731c4828a47-0
01:13:55.320 --> 01:13:58.524
I'm not able to recollect it but

it it's a bit faster if you use

5fcea510-7d01-437c-b67e-2731c4828a47-1
01:13:58.524 --> 01:13:59.560
it in the aggregator.

c25c7b3d-9efc-49da-b311-c2c720638542-0
01:14:00.480 --> 01:14:01.400
So I don't remember it.

6e57d771-a5c9-4552-8237-6642949fb7fe-0
01:14:01.400 --> 01:14:02.880
So I'll, I'll check it out and
let you know.

e99f7994-f8da-4e07-ae2f-4ba67b4eca43-0
01:14:03.440 --> 01:14:07.720
So am I doing?

210e640d-2251-42c5-9448-edb37f294617-0
01:14:07.720 --> 01:14:10.640
Yeah, it's it's getting, yeah,
saved, right?

3b79e795-b4e9-4291-956b-30ced4ca2cc1-0
01:14:10.640 --> 01:14:15.760
So yeah, so let me clean the
console.

d8478605-6395-4e41-818c-45dfd8596c41-0
01:14:18.080 --> 01:14:25.960
OK, so OK, so let's go and make
the same 10 points debug.

e2b3bada-4048-49f9-bd7d-dba76268080d-0
01:14:26.040 --> 01:14:27.160
What is the debug mode?

61ac773a-a135-4d60-a548-18bcbb764348-0
01:14:30.160 --> 01:14:30.360
Right.

23fd7c40-cb95-4c24-97ef-f9c5149ae013-0
01:14:30.360 --> 01:14:33.300
So if you see it's going to
process all the records, then it

23fd7c40-cb95-4c24-97ef-f9c5149ae013-1
01:14:33.300 --> 01:14:36.000
comes to the aggregator and it
does the streaming part.

f5c88f20-5643-4d8d-881c-b5cab749aca2-0
01:14:36.320 --> 01:14:39.580
So here you should be seeing a
valid Jason, not like the

f5c88f20-5643-4d8d-881c-b5cab749aca2-1
01:14:39.580 --> 01:14:42.040
earlier one which had two
separate arrays.

7d25fba9-3ecd-41e3-ab01-7a586efc3e3e-0
01:14:42.560 --> 01:14:47.096
OK so the so if you ask like

when to use streaming and when

7d25fba9-3ecd-41e3-ab01-7a586efc3e3e-1
01:14:47.096 --> 01:14:50.120
to use the batch size,
aggregator size.

9c34df02-7b68-42da-8a6c-7fbff7af96ae-0
01:14:50.360 --> 01:14:53.990
So use aggregator size if you
are inserting it to a database

9c34df02-7b68-42da-8a6c-7fbff7af96ae-1
01:14:53.990 --> 01:14:57.502
or to a sales force account or
and use streaming if you're

9c34df02-7b68-42da-8a6c-7fbff7af96ae-2
01:14:57.502 --> 01:15:00.954
trying to you know load it to
some file based system like

9c34df02-7b68-42da-8a6c-7fbff7af96ae-3
01:15:00.954 --> 01:15:02.800
FTPS, FTP or in the local pile.

8f031ae8-1244-425d-af78-a7946c13afa5-0
01:15:04.440 --> 01:15:06.960
Yes, any question before I go to
the last topic.

de35e4d0-540f-420b-a50f-f91067c5867c-0
01:15:13.040 --> 01:15:17.320
I think I might say OK so let's
go to the last topic.

2ace7689-ff14-418e-9f94-3de5e16d29f4-0
01:15:17.320 --> 01:15:20.870
So in the last topic I want to
talk about what happens if a

2ace7689-ff14-418e-9f94-3de5e16d29f4-1
01:15:20.870 --> 01:15:21.640
record fails.

d04a8e80-5a8e-4d74-b893-32e501fc51e4-0
01:15:22.120 --> 01:15:26.368
OK so let me remove all the
break points or let me add one

d04a8e80-5a8e-4d74-b893-32e501fc51e4-1
01:15:26.368 --> 01:15:28.240
break point here only one.

2f424ce4-1167-47a8-980e-baa9b0354cd0-0
01:15:28.640 --> 01:15:33.165
So in the so if you remember our
transform message it is going to

2f424ce4-1167-47a8-980e-baa9b0354cd0-1
01:15:33.165 --> 01:15:37.417
give you true or false based on
the ID which is being sent in

2f424ce4-1167-47a8-980e-baa9b0354cd0-2

01:15:37.417 --> 01:15:38.240
the request.

766f62ca-24be-4d0d-904d-7539a2bda02b-0
01:15:38.400 --> 01:15:40.040
So it has to be a integer.

6ac816a4-ef9a-49a9-8cbe-a4eb385edd7d-0
01:15:40.400 --> 01:15:44.672
So in some cases because of some
human error or because of you

6ac816a4-ef9a-49a9-8cbe-a4eb385edd7d-1
01:15:44.672 --> 01:15:48.673
know mistake in the logic,
instead of an integer you might

6ac816a4-ef9a-49a9-8cbe-a4eb385edd7d-2
01:15:48.673 --> 01:15:50.640
get a string for the ID part.

c7e546c0-2059-47ea-b79b-7a16c7d52e7f-0
01:15:50.840 --> 01:15:53.774
OK, So what happens is it's it's
it's going to of course break

c7e546c0-2059-47ea-b79b-7a16c7d52e7f-1
01:15:53.774 --> 01:15:54.520
the application.

e362b636-c68c-4a26-9c46-9fff7704049c-0
01:15:55.080 --> 01:15:59.680
So in batch processing if you
see here there is an error.

72857469-289f-4e90-b374-f8f5adff7222-0
01:15:59.800 --> 01:16:02.378
So the error is basically that
you know you need to have a

72857469-289f-4e90-b374-f8f5adff7222-1
01:16:02.378 --> 01:16:04.040
integer but you are getting a
string.

275155b2-b0a5-4aba-9688-4daaae6ced12-0
01:16:04.040 --> 01:16:07.208
OK, it expects both number and
number, but we are getting in a

275155b2-b0a5-4aba-9688-4daaae6ced12-1
01:16:07.208 --> 01:16:07.560
string.

3a2030ed-ec05-4407-acf9-cc3c38668081-0
01:16:08.080 --> 01:16:12.109
So what happens now is in the
default behavior, the batch step

3a2030ed-ec05-4407-acf9-cc3c38668081-1
01:16:12.109 --> 01:16:16.202
where the error occurred, it's
going to continue processing the

3a2030ed-ec05-4407-acf9-cc3c38668081-2

01:16:16.202 --> 01:16:19.080
remaining records which are
there in memory.

1b71d44d-bf40-4334-beca-82e403aa0919-0
01:16:19.440 --> 01:16:23.176
OK, so the records which are
there in the in memory, so those

1b71d44d-bf40-4334-beca-82e403aa0919-1
01:16:23.176 --> 01:16:26.792
records will continue processing
and it's going to stop the

1b71d44d-bf40-4334-beca-82e403aa0919-2
01:16:26.792 --> 01:16:28.600
processing in step one itself.

b00839be-dbe2-4e04-8f76-474a94ed330e-0
01:16:28.840 --> 01:16:31.480
The Step 2 will not be executed.

3afe2d26-9a56-432b-943e-faf1f5d911e5-0
01:16:31.880 --> 01:16:33.880
So that is the default behavior.

55bdc835-3d53-4fab-8c15-b3bf2d89a873-0
01:16:34.160 --> 01:16:38.948
And if you see the console, it's
going to show in which step

55bdc835-3d53-4fab-8c15-b3bf2d89a873-1
01:16:38.948 --> 01:16:43.972
there was an error or exception
and how many records are failed

55bdc835-3d53-4fab-8c15-b3bf2d89a873-2
01:16:43.972 --> 01:16:46.720
or how many records had the
issue.

4a847860-209b-4f42-9625-503e8e429d6d-0
01:16:47.280 --> 01:16:51.200
OK, so that is automatically
logged by the on complete phase.

15c2de6a-405f-457c-8806-07dc28e432bc-0
01:16:51.640 --> 01:16:55.538
And again, this is kind of a
downside or limitation, because

15c2de6a-405f-457c-8806-07dc28e432bc-1
01:16:55.538 --> 01:16:59.372
if you're working with millions
of records, and if some 100

15c2de6a-405f-457c-8806-07dc28e432bc-2
01:16:59.372 --> 01:17:03.079
records failed or 100 records
have some invalid, you know

15c2de6a-405f-457c-8806-07dc28e432bc-3
01:17:03.079 --> 01:17:07.041
data, you don't want to stop
processing the remaining 900,000

15c2de6a-405f-457c-8806-07dc28e432bc-4
01:17:07.041 --> 01:17:08.000
records, right?

daa98acf-2894-4b40-a430-f8ae64f0b257-0
01:17:08.280 --> 01:17:12.800
So what you can do is you can,
you know, define a limit.

39558a45-d441-44b9-bbd1-ea460bf26567-0
01:17:12.800 --> 01:17:15.240
The limit is Max failed records.

ea2ef8e8-b554-45a0-a551-0b2ef2a75b51-0
01:17:15.440 --> 01:17:18.861
By default it is 0, which means
that even if one record failed,

ea2ef8e8-b554-45a0-a551-0b2ef2a75b51-1
01:17:18.861 --> 01:17:21.160
the next batch steps will not be
executed.

90032940-3343-4cce-a95b-ed52d0040fce-0
01:17:21.440 --> 01:17:22.520
So you can give 100.

64830477-bb52-482f-b42d-de20b134086c-0
01:17:22.520 --> 01:17:25.958
If you give hundred, it's going
to allow up to 100 failed

64830477-bb52-482f-b42d-de20b134086c-1
01:17:25.958 --> 01:17:29.694
records and the next 101 record
which is failed, it's going to

64830477-bb52-482f-b42d-de20b134086c-2
01:17:29.694 --> 01:17:30.879
stop the processing.

d4737648-80ab-4e95-baf2-48976f7b8859-0
01:17:31.200 --> 01:17:35.499
If you don't want to have you
know any limit, you can use -1

d4737648-80ab-4e95-baf2-48976f7b8859-1
01:17:35.499 --> 01:17:35.640
-.

afb198e8-2df3-49aa-9245-50fb0ed3a0dc-0
01:17:35.640 --> 01:17:37.840
1 is unlimited number of failed
records.

e63cfa4a-63ab-4abd-86ca-944ce2174821-0
01:17:38.560 --> 01:17:39.880
OK, OK, I'll.

292ee1ee-3149-4c00-b54c-4e0e6912675b-0
01:17:40.200 --> 01:17:41.360
I'll just use ten and seven.

78b3c3b7-323e-4171-9415-ed5e37f3ebc1-0

01:17:41.480 --> 01:17:45.200
Just go ahead and just a step
back see that.

94a92a1d-609f-4bab-a852-1f856b39b4cd-0
01:17:45.560 --> 01:17:49.313
So if you said it's like it
stops at, you don't process

94a92a1d-609f-4bab-a852-1f856b39b4cd-1
01:17:49.313 --> 01:17:50.520
further on Step 2.

67c6def1-e5ad-4500-a68f-100d2231747f-0
01:17:50.560 --> 01:17:52.680
I mean that's we have two batch
2 steps, right?

64ee34ab-d2a9-44ee-b3a1-6d3a6c3be46a-0
01:17:52.680 --> 01:17:54.400
Batch steps to stop there.

d6ef9169-ca9a-4f7e-9f98-b0b5c216fc54-0
01:17:54.400 --> 01:17:57.567
But the the the persistent
queue, it is still when you

d6ef9169-ca9a-4f7e-9f98-b0b5c216fc54-1
01:17:57.567 --> 01:17:59.640
it'll be there or like how is
that?

bf26fa23-548d-4bdd-a3e0-fb814f231bcb-0
01:18:00.240 --> 01:18:03.589
I think it won't be since it's a
good question since it is, it is

bf26fa23-548d-4bdd-a3e0-fb814f231bcb-1
01:18:03.589 --> 01:18:04.960
still not completed, right?

3afd9f2e-fc1e-428a-8980-ddb7f74020df-0
01:18:04.960 --> 01:18:06.960
It should delete only after
completion of it, correct?

d1952a35-bdf0-47a3-a660-1a1befd4c0fe-0
01:18:06.960 --> 01:18:08.960
No, I think it it, it entered
the error state, right?

6d5c1bde-d1b3-4e99-a467-eb5843a0abde-0
01:18:08.960 --> 01:18:09.920
So I think I error state.

239fc9ff-7310-42e7-8a3b-d879fa2fb4b7-0
01:18:09.920 --> 01:18:10.040
Yeah.

dc3dab57-41c6-4d9e-835e-1fecfce0bd1a-0
01:18:10.040 --> 01:18:11.960
OK, let me see.

f5dff19f-9759-4286-a7ed-8af9479b8a1f-0
01:18:14.600 --> 01:18:15.880

Let's go to the queues.

f8f7a903-4aaf-4814-ab97-30bf1f59b9b1-0
01:18:17.360 --> 01:18:19.400
Yeah, I see it's going to delete
the queues as well.

8adc41f9-1c33-4abf-b943-17bf77dcd2a8-0
01:18:19.400 --> 01:18:22.809
If there is an error state, the
default behavior is it's going

8adc41f9-1c33-4abf-b943-17bf77dcd2a8-1
01:18:22.809 --> 01:18:25.840
to just stop the processing and
also delete the queues.

cf0a3149-a0c5-49c1-a33d-08a6d944946c-0
01:18:26.360 --> 01:18:30.373
OK, so to avoid this you have to
always make use of this Max

cf0a3149-a0c5-49c1-a33d-08a6d944946c-1
01:18:30.373 --> 01:18:31.360
failed records.

647eaa20-152f-4134-95ef-8d77efad78a7-0
01:18:31.560 --> 01:18:36.320
OK, so I'm just using 10 as an
example, so let's test this out.

b71f1065-57e3-423a-8406-45f33ac2a8b0-0
01:18:39.400 --> 01:18:40.400
Go to.

ac3498b0-ca6b-4e3a-8086-330217539adc-0
01:18:41.560 --> 01:18:47.720
Yeah, so the 6th record failed
of course.

06ac8b3e-c16e-4539-a5fc-3c8d57e74be6-0
01:18:47.720 --> 01:18:49.360
So you see an error here.

c0aa6b06-3ab4-4c53-b692-d4a271de5b98-0
01:18:49.720 --> 01:18:53.040
But let's see what happens now I
have to consume them.

c5a6e7fe-6247-4398-8fc2-0589ebed29f9-0
01:18:54.200 --> 01:18:56.960
So if you see, the second step
is getting executed.

fe9e9c05-19c9-49be-b345-f27714582e25-0
01:18:57.120 --> 01:19:01.848
The reason is this batch job can
now accept and up to 10 failed

fe9e9c05-19c9-49be-b345-f27714582e25-1
01:19:01.848 --> 01:19:02.440
records.

0adb1886-eeeb-46f8-b3d7-44bfd66b5746-0
01:19:02.440 --> 01:19:04.847

The 11th record which is failed,
it's going to stop the

0adb1886-eeeb-46f8-b3d7-44bfd66b5746-1
01:19:04.847 --> 01:19:05.320
processing.

03e0bbea-9e8c-4ced-8d26-cd83b05810a8-0
01:19:05.520 --> 01:19:08.480
OK, so that's how the Max failed
record works.

928a9995-dc48-4d89-b9df-c5d121827529-0
01:19:08.760 --> 01:19:13.237
OK, so I think if I go to Jason
again, it's going to it's it's

928a9995-dc48-4d89-b9df-c5d121827529-1
01:19:13.237 --> 01:19:17.360
not going to have the 6th field,
you only see 248 and 10.

db100221-87de-4b66-8d44-bb66cce55a02-0
01:19:17.760 --> 01:19:20.240
OK, so you might get a question.

f0ab5cb6-2641-481f-9c1a-712e02d4b4a5-0
01:19:20.360 --> 01:19:24.091
So, so that I want to know which
record failed because I need to

f0ab5cb6-2641-481f-9c1a-712e02d4b4a5-1
01:19:24.091 --> 01:19:26.560
troubleshoot why the record
failed, right.

f4bce584-91a7-4b4e-abe0-50a83376dc35-0
01:19:26.560 --> 01:19:30.071
So you need to also have a
process to store or to identify

f4bce584-91a7-4b4e-abe0-50a83376dc35-1
01:19:30.071 --> 01:19:31.560
the records which failed.

ecd3161f-591f-432a-b002-5c7e041cdcd3-0
01:19:31.840 --> 01:19:35.288
So batch job automatically
identifies the records which

ecd3161f-591f-432a-b002-5c7e041cdcd3-1
01:19:35.288 --> 01:19:35.720
failed.

216c2ec9-899d-437e-beaa-172c1b376e39-0
01:19:35.880 --> 01:19:38.400
So you have to, you know,
process them.

7262e2ea-8bcf-4d85-90e4-f2af581e0b83-0
01:19:38.400 --> 01:19:42.400
So how do you identify and
record the failed records?

4538937a-ebf5-4168-b9d1-f29c72afcf4a-0

01:19:43.000 --> 01:19:45.760
So again you can rely on another
batch step.

1e58026d-8fcd-4c15-bcd5-7b8470f22f59-0
01:19:47.840 --> 01:19:52.920
I'll just say logging failed
records batch step and you can

1e58026d-8fcd-4c15-bcd5-7b8470f22f59-1
01:19:52.920 --> 01:19:57.831
use logger or you can use a
database insert or a queue so

1e58026d-8fcd-4c15-bcd5-7b8470f22f59-2
01:19:57.831 --> 01:20:03.249
that you can process this in an
external system which has to be

1e58026d-8fcd-4c15-bcd5-7b8470f22f59-3
01:20:03.249 --> 01:20:04.519
troubleshooted.

1dad701f-ed55-448a-91bd-7f99e5d2ea36-0
01:20:04.920 --> 01:20:08.720
I'll just use a payload, OK?

2bb492bd-25d1-4c44-b763-74ade943aaff-0
01:20:09.200 --> 01:20:12.800
So I'll remove the break point
and add the break point here.

cc99b866-6895-4cc5-9d4c-a87745997ba0-0
01:20:13.040 --> 01:20:16.540
But here in this step I only
want to process the failed

cc99b866-6895-4cc5-9d4c-a87745997ba0-1
01:20:16.540 --> 01:20:17.040
records.

2f08edbb-a872-4bf6-9e52-3de6a29b05a6-0
01:20:17.240 --> 01:20:19.200
I don't want to process the
successful records.

43367ce0-79f5-4cea-92b8-777331a039c3-0
01:20:19.360 --> 01:20:21.760
How do I do that if I click on
the batch step?

09b96224-9886-4058-87d3-5d08409004ab-0
01:20:22.120 --> 01:20:25.748
We used accept expression
earlier, but now we'll use the

09b96224-9886-4058-87d3-5d08409004ab-1
01:20:25.748 --> 01:20:26.640
accept policy.

c3a6d651-f257-4fff-a25c-6866dbaa8e71-0
01:20:26.880 --> 01:20:28.000
What is this policy?

123f33b5-07ac-40c1-8782-b98f0b5dc5bb-0

01:20:28.240 --> 01:20:30.440
So again, we have three types of
policies.

94150e55-341f-41e8-b4cf-ef6df4ed395e-0
01:20:30.440 --> 01:20:34.549
By default the policy is NO
failure, which means that the

94150e55-341f-41e8-b4cf-ef6df4ed395e-1
01:20:34.549 --> 01:20:39.013
batch step will only execute the
records which did not fail in

94150e55-341f-41e8-b4cf-ef6df4ed395e-2
01:20:39.013 --> 01:20:40.360
the previous steps.

a3c1f3a8-f97a-4606-9e2a-d6ef6beac2a9-0
01:20:40.760 --> 01:20:42.040
So that is no failures.

d4a68e4e-2bea-4a93-8e9e-144db7f87e13-0
01:20:42.400 --> 01:20:44.080
You have only failures.

b6faa6ec-735b-4cb2-a834-8500be69317d-0
01:20:44.200 --> 01:20:48.939
Which means that this batch step
is going to you know process the

b6faa6ec-735b-4cb2-a834-8500be69317d-1
01:20:48.939 --> 01:20:53.534
records which have failed in the
previous batch steps, only the

b6faa6ec-735b-4cb2-a834-8500be69317d-2
01:20:53.534 --> 01:20:58.130
failed records OK and again the
status of the record successful

b6faa6ec-735b-4cb2-a834-8500be69317d-3
01:20:58.130 --> 01:20:58.920
or failure.

f9deb8a1-fbb6-4a39-b407-7bb39e3ab9f1-0
01:20:59.080 --> 01:21:02.342
It is stored in the you know
records metadata so that is

f9deb8a1-fbb6-4a39-b407-7bb39e3ab9f1-1
01:21:02.342 --> 01:21:05.834
automatically stored in the
metadata so that will be fetched

f9deb8a1-fbb6-4a39-b407-7bb39e3ab9f1-2
01:21:05.834 --> 01:21:09.039
automatically and it will be
checked for processing the

f9deb8a1-fbb6-4a39-b407-7bb39e3ab9f1-3
01:21:09.039 --> 01:21:09.440
record.

af28a8fd-76dc-48e1-82fc-b6f757961203-0

01:21:09.680 --> 01:21:13.051
And finally the policy you have
all all means this batch step

af28a8fd-76dc-48e1-82fc-b6f757961203-1
01:21:13.051 --> 01:21:16.640
will process both failed records
and the success records as well.

c397c052-bd8b-458c-ba2a-649c0736d0e6-0
01:21:17.120 --> 01:21:20.440
OK, so that is how the accept
policy works.

800e994c-9748-4dc6-9f5a-bd5ac2cc1b6f-0
01:21:21.680 --> 01:21:23.120
So I saved the application
already.

670edf53-d0b5-479d-9a7c-fcf72f30d368-0
01:21:23.120 --> 01:21:26.040
So let me oh, it's again getting
saved.

6d3c772f-accd-4f96-99d7-44969c758933-0
01:21:26.040 --> 01:21:27.480
OK, it's still getting deployed.

f4c886db-1584-4e37-82d7-9e7e048836c6-0
01:21:32.680 --> 01:21:33.560
It's redeployed.

68fd616d-a3b5-4e22-ad64-be74fe4c9ac8-0
01:21:33.640 --> 01:21:37.400
So now let's do the last one if
I make a call.

4137865f-3fc6-4edc-9653-ee088fb1a650-0
01:21:37.960 --> 01:21:40.849
So ideally speaking, this record
should be logged in the final

4137865f-3fc6-4edc-9653-ee088fb1a650-1
01:21:40.849 --> 01:21:42.960
batch step because this is the
failed record.

45a7e6ae-c293-4425-8554-34a6ed203df3-0
01:21:43.040 --> 01:21:44.720
OK, so let's do next.

3855b165-0f3c-4ea0-be09-053510bd3edf-0
01:21:46.760 --> 01:21:48.680
OK, it's failing in the first
batch step.

3b304ee4-2283-40be-8be9-9e198778e595-0
01:21:48.680 --> 01:21:51.040
So let me just consume the first
batch step.

15d5e72d-6f5e-41dc-9783-c06b50d36503-0
01:21:53.400 --> 01:21:57.200
It's going to do the second step
and now if you see it came

15d5e72d-6f5e-41dc-9783-c06b50d36503-1
01:21:57.200 --> 01:22:00.240
briefly to the third step by I
just skipped it.

f7a4b173-727c-4d31-adb6-9aab5f78ba04-0
01:22:00.400 --> 01:22:03.441
So if you see here, it's also
going to log the 6th record

f7a4b173-727c-4d31-adb6-9aab5f78ba04-1
01:22:03.441 --> 01:22:05.120
which was failed using a logger.

6ccb6f10-bff1-47c5-8d87-cae2f2ba3224-0
01:22:05.400 --> 01:22:08.346
So it's up to you like how you
want to, you know, work with

6ccb6f10-bff1-47c5-8d87-cae2f2ba3224-1
01:22:08.346 --> 01:22:09.280
this failed record.

4f3bd8f7-1552-4129-ba6c-164275e5aa11-0
01:22:09.680 --> 01:22:12.610
OK, so once the batch step is
completed, of course the records

4f3bd8f7-1552-4129-ba6c-164275e5aa11-1
01:22:12.610 --> 01:22:15.262
will be deleted from the VM
queues as well the persisted

4f3bd8f7-1552-4129-ba6c-164275e5aa11-2
01:22:15.262 --> 01:22:15.960
queues as well.

0230c557-7da8-4f57-b059-e098d56fef9e-0
01:22:16.560 --> 01:22:20.930
OK, And if you Scroll down, OK,
so if you Scroll down, it's

0230c557-7da8-4f57-b059-e098d56fef9e-1
01:22:20.930 --> 01:22:25.592
going to also show you that you
know it processed 10 records, 9

0230c557-7da8-4f57-b059-e098d56fef9e-2
01:22:25.592 --> 01:22:28.360
are successful and one record
failed.

a2311075-e96d-4d14-bbc4-523f086acdf5-0
01:22:28.440 --> 01:22:31.361
So that is you're getting this
log automatically from the on

a2311075-e96d-4d14-bbc4-523f086acdf5-1
01:22:31.361 --> 01:22:32.080
complete phase.

3c179246-109f-4650-b89b-39637989b87d-0
01:22:33.400 --> 01:22:33.760
Yep.

f96ff832-198a-4b80-9b01-7db1c44f51c9-0
01:22:33.880 --> 01:22:34.760
Any questions?

30843d86-8222-41d7-bd82-b7757cb001dc-0
01:22:39.760 --> 01:22:40.480
Not from my side.

59f4150e-17a1-481c-85c2-ce6d8df41390-0
01:22:40.520 --> 01:22:40.760
Yeah.

1007a6e2-6ca0-4498-b573-8d8ae3130a98-0
01:22:41.560 --> 01:22:44.560
And Alice, Dinesh, Raish.

478cc19f-4b65-49c2-9ae1-30f9a01a06ea-0
01:22:44.560 --> 01:22:49.040
OK, so if I come here, I think
we covered.

0981b3b7-d5b5-4e34-a19a-653c49ab14ed-0
01:22:49.040 --> 01:22:50.160
Fail looks fine.

29d110ed-20ab-4f7d-9237-18a612078c77-0
01:22:50.240 --> 01:22:50.920
Yeah, it looks fine.

325fe2b8-3e9e-460a-8243-4316f71165f2-0
01:22:51.440 --> 01:22:54.760
Yeah yeah, that's great.

0ca8963e-ca74-4b84-88fd-9db04a871194-0
01:22:55.680 --> 01:22:57.320
So I think one last thing is
left.

d8effe69-4acc-4bb6-8545-1f19f06d5f7c-0
01:22:57.320 --> 01:22:58.320
We have 7 minutes.

68cdb348-73e8-42af-9aae-ecb08c7e0258-0
01:22:58.320 --> 01:22:59.480
I'll cover that one as well.

c7fed61c-ea8c-4371-91ab-190dd0c008d9-0
01:22:59.480 --> 01:23:00.120
Variables.

d284cac8-ae2b-4819-8e69-d9c1567a1198-0
01:23:00.120 --> 01:23:03.800
So how does the variables behave
in batch job?

5b127e8c-a811-4395-bd65-f723a3274728-0
01:23:04.360 --> 01:23:07.480
And it is again a bit weird, so
let me explain why.

5b3120c3-473a-420c-b5bd-62b6cb9a85ea-0
01:23:08.440 --> 01:23:11.960
So I'll use a different batch
job.

a791cf98-8081-40e9-8825-6a363c9d5abd-0
01:23:12.000 --> 01:23:18.518
I don't want to make changes
there, so this job I just want

a791cf98-8081-40e9-8825-6a363c9d5abd-1
01:23:18.518 --> 01:23:21.560
to use the path as variable.

1e44d503-7f53-4ea7-aaf0-0fa91d741c25-0
01:23:22.120 --> 01:23:24.800
OK, I'll use one batch job.

e5225cee-2e80-467f-a8cb-bf337a6c936b-0
01:23:26.320 --> 01:23:29.960
Outside the batch job I want to
use SET variable.

851c503d-036f-41cf-a209-0034cfba1c4f-0
01:23:31.520 --> 01:23:41.200
OK, so the name of the variable,
let's say it is code code 12345.

ee890dfe-6050-4cad-ba50-b3660e39d419-0
01:23:41.400 --> 01:23:43.480
OK, so I'll just say 12345.

9485ab0d-a0e2-4711-9995-1d7ec6f35ffc-0
01:23:43.640 --> 01:23:46.160
I'm just setting a variable code
is equal to 12345.

26ab7e62-442a-4950-8237-7a9f984119e8-0
01:23:46.280 --> 01:23:52.012
The same variable I want to set
inside but inside I want to use

26ab7e62-442a-4950-8237-7a9f984119e8-1
01:23:52.012 --> 01:23:54.520
ABCD or you know ABCD batch.

e28c409a-62ec-48aa-af28-63163699aceb-0
01:23:55.320 --> 01:24:00.003
OK, so now I'm setting up this
variable here, so let's see how

e28c409a-62ec-48aa-af28-63163699aceb-1
01:24:00.003 --> 01:24:02.680
I can access this variables
inside.

b27c5041-ec22-47a2-94d1-02a8415c382a-0
01:24:02.800 --> 01:24:06.968
OK, so if I use a logger, but
you're given the same name,

b27c5041-ec22-47a2-94d1-02a8415c382a-1
01:24:06.968 --> 01:24:07.400
right?

154caf0a-fc13-472f-a72a-0ed299aa88e4-0
01:24:07.640 --> 01:24:07.840
Both.

bbcb76da-4455-48f4-bd1c-ae520ddc78b3-0
01:24:07.960 --> 01:24:09.080
Yeah, same name use.

5cd41930-d0e5-4a23-8216-4fe8397fcd15-0
01:24:09.240 --> 01:24:10.080
I'm using the same name.

0feb2638-3272-4a95-a6cf-81bac41f174c-0
01:24:10.080 --> 01:24:11.240
I want to show you how it works.

ffd2928e-f198-40a2-965a-291b249bca8f-0
01:24:11.600 --> 01:24:16.160
So first I want to say I want to
log the vas dot code.

b4c68f50-7a34-4915-a6a2-6feb3aaf858c-0
01:24:16.280 --> 01:24:19.280
OK so just copy this.

ee8c87c3-9dba-4ead-bb3b-41c0252509f1-0
01:24:19.800 --> 01:24:22.680
Yeah so let's see what will be
logged here.

b170d716-095c-43d9-b4b9-7b0f8c379dc8-0
01:24:23.760 --> 01:24:27.800
Next I want to add another batch
step.

0ebb3722-4002-434d-8e08-c996ae9616a6-0
01:24:29.160 --> 01:24:32.332
So I want to log the same
variable here as well, and I

0ebb3722-4002-434d-8e08-c996ae9616a6-1
01:24:32.332 --> 01:24:34.120
want to use aggregator as well.

2d464a7a-e23e-44bf-aaf9-ca9e4a88b569-0
01:24:34.520 --> 01:24:37.400
And I want to see how the
variable works here.

97c91e0b-71f5-415e-bd6a-8b0e90ca5d68-0
01:24:37.600 --> 01:24:41.937
I want to see how the variable
is logged in the on complete and

97c91e0b-71f5-415e-bd6a-8b0e90ca5d68-1
01:24:41.937 --> 01:24:42.480
outside.

00dbad29-8f87-4e52-a3a2-5736336b8b2d-0
01:24:43.320 --> 01:24:44.320
OK, sorry, Outside.

31a841a1-ec6a-4343-9c8e-f72b3f8c49e6-0
01:24:45.520 --> 01:24:52.120
OK, yeah, so OK Aggregate AI
need to give aggregate a size.

c29c5062-04d6-4931-bf9a-3a1317f9937f-0
01:24:52.120 --> 01:24:54.724

If you don't give aggregate a
size, the application will not

c29c5062-04d6-4931-bf9a-3a1317f9937f-1
01:24:54.724 --> 01:24:56.560
deploy, it's going to fail the
deployment.

ba1c8e37-a839-480f-874a-579366c0aaf1-0
01:24:58.080 --> 01:25:02.648
OK so let's add a break point to
all the things and see how they

ba1c8e37-a839-480f-874a-579366c0aaf1-1
01:25:02.648 --> 01:25:03.000
work.

2e70cc5f-ef20-4f00-b4bf-ea755f96d1ff-0
01:25:03.760 --> 01:25:07.644
So the the The thing is whatever
variables which we are setting

2e70cc5f-ef20-4f00-b4bf-ea755f96d1ff-1
01:25:07.644 --> 01:25:11.406
in the batch step, those are
only accessible in another batch

2e70cc5f-ef20-4f00-b4bf-ea755f96d1ff-2
01:25:11.406 --> 01:25:12.559
step process phase.

d916afe9-001d-4551-a77f-d77693bcc494-0
01:25:12.720 --> 01:25:17.840
You cannot use this variable in
aggregator or in on complete or

d916afe9-001d-4551-a77f-d77693bcc494-1
01:25:17.840 --> 01:25:19.600
outside the batch job.

adcd905b-3d85-4489-9498-5c0fbfbe568d-0
01:25:19.600 --> 01:25:23.171
Also you cannot access the
variables which you are setting

adcd905b-3d85-4489-9498-5c0fbfbe568d-1
01:25:23.171 --> 01:25:24.200
in the batch tab.

238c5fa6-a5f7-48d1-a7fb-9bce408a0fec-0
01:25:24.360 --> 01:25:25.520
So that is the concept.

eecd3136-c3cd-4403-ad5f-f6594c9965e4-0
01:25:25.760 --> 01:25:29.040
So I want to show you like how
it works.

54a811db-00f1-4cc9-b3c0-8279cd110eb1-0
01:25:29.120 --> 01:25:31.360
OK, so OK, it's still loading.

174f22b8-5096-4ebe-a40d-3367df6e18d6-0
01:25:34.960 --> 01:25:35.800
Yeah, that's it.

5f7d35cc-ded9-4bba-ae94-a18640f68c3f-0
01:25:36.320 --> 01:25:42.640
So let's make a call to use a
simple variable.

a22bf8bd-ca3f-4790-8934-00deb9782ec0-0
01:25:43.440 --> 01:25:51.760
OK yeah, so now let me clean the
console if I scroll.

c5439d1f-6eb0-431f-b60c-19f2d398c10b-0
01:25:52.280 --> 01:25:57.859
So whatever variable which we
are setting outside the batch

c5439d1f-6eb0-431f-b60c-19f2d398c10b-1
01:25:57.859 --> 01:25:59.440
job that one SEC.

5940b2f8-0187-44de-8c30-03dce1542734-0
01:26:00.520 --> 01:26:04.160
So I guess that will be you know
accessible in this logger.

3b68708e-f2a0-40e0-b48c-97b02112cfd0-0
01:26:04.400 --> 01:26:05.840
OK which is there in the batch
step.

814efca3-0999-4d87-82d3-b4bb2684b37b-0
01:26:06.120 --> 01:26:09.360
And now in the batch step I am
setting another variable.

18b96327-f647-4214-8688-1bcc1e7d236d-0
01:26:09.800 --> 01:26:12.716
OK so if you see the code is
available inside but in batch

18b96327-f647-4214-8688-1bcc1e7d236d-1
01:26:12.716 --> 01:26:15.040
step I'm setting this variable
called as ABCD.

cafb88bb-f228-4af6-905a-bd75f71de0ef-0
01:26:16.840 --> 01:26:18.640
OK, so this will be executable.

ff526814-44fd-44e2-a85a-1da40d9af0a8-0
01:26:18.640 --> 01:26:20.120
OK just give me one second.

e0a00675-05d7-479c-b785-e71592cda3eb-0
01:26:20.720 --> 01:26:23.840
I just want to I just want to
work with one record.

d6e1325b-70e1-430d-ae8c-2509093012e1-0
01:26:23.840 --> 01:26:26.920
OK, it won't make sense if you
have 10 records.

855b7f2c-361e-4e73-815a-27147f7e7553-0
01:26:30.000 --> 01:26:33.143

OK, so let me clean this,
because I have to do it 10

855b7f2c-361e-4e73-815a-27147f7e7553-1
01:26:33.143 --> 01:26:35.160
times, you know, for each
record.

1ae5930a-424c-47ea-ad07-53ecebf5362c-0
01:26:35.760 --> 01:26:37.400
So let me remove all these
things.

37a05f5a-3524-4033-857c-d00ebca8c4b4-0
01:26:40.440 --> 01:26:42.040
Well, let's just use one record.

16a40d0e-feb9-48ef-a356-b32dbdc19eef-0
01:26:43.880 --> 01:26:44.240
OK?

01bb5b62-9b1b-453a-a3d7-3d7a288d351a-0
01:26:44.720 --> 01:26:45.200
Clean.

7e2dde55-abc5-4adb-ac3f-680a59b3e173-0
01:26:46.640 --> 01:26:51.240
OK Yeah 123 it's going to set
this variable.

d5d1ec47-af04-44d1-92cd-e25dd781818b-0
01:26:51.240 --> 01:26:54.370
So now the variable value will
be ABCD batch because this we

d5d1ec47-af04-44d1-92cd-e25dd781818b-1
01:26:54.370 --> 01:26:56.320
have set it up inside the batch
step.

4412acba-8ae0-4e76-9c44-e02625895d8f-0
01:26:56.520 --> 01:27:00.352
So now I can access this
variable within the next batch

4412acba-8ae0-4e76-9c44-e02625895d8f-1
01:27:00.352 --> 01:27:04.389
step only in the process phase
not in the batch aggregator

4412acba-8ae0-4e76-9c44-e02625895d8f-2
01:27:04.389 --> 01:27:04.800
phase.

a17e300b-e2d0-4260-a89c-0e320da5c4d1-0
01:27:05.000 --> 01:27:11.144
So if you see I can log this so
it should log, if you see it

a17e300b-e2d0-4260-a89c-0e320da5c4d1-1
01:27:11.144 --> 01:27:13.360
should log abcd batch.

ebbcc232-f930-4695-8766-b6d90cc7399f-0
01:27:13.560 --> 01:27:15.640

But this logger should not work.

a57d3b2f-edbd-41bd-befc-05821e754784-0
01:27:15.640 --> 01:27:19.186
So ideally speaking if I do
next, it's just going to use

a57d3b2f-edbd-41bd-befc-05821e754784-1
01:27:19.186 --> 01:27:19.560
12345.

6d6b0b50-598c-4b4f-9a49-3fb57de16119-0
01:27:19.560 --> 01:27:22.720
This is the variable which we
set outside the batch job.

794288a4-9b51-499b-a3d6-fad0f637b75a-0
01:27:23.280 --> 01:27:27.347
So this ABCD batch, this cannot
be accessed in the batch

794288a4-9b51-499b-a3d6-fad0f637b75a-1
01:27:27.347 --> 01:27:28.560
aggregator right?

072f6a28-e1b1-462b-b751-9d9b6ea8891d-0
01:27:28.560 --> 01:27:31.640
So if I come to on complete
phase again the on complete

072f6a28-e1b1-462b-b751-9d9b6ea8891d-1
01:27:31.640 --> 01:27:34.775
phase it cannot access the
variable which we have set in

072f6a28-e1b1-462b-b751-9d9b6ea8891d-2
01:27:34.775 --> 01:27:35.600
the batch step.

89f756b3-c338-4e56-a9ed-0f4f5e48ed12-0
01:27:35.840 --> 01:27:40.320
So it should again log 12345
which we have set outside the

89f756b3-c338-4e56-a9ed-0f4f5e48ed12-1
01:27:40.320 --> 01:27:41.080
batch job.

c9d9d536-b45d-4491-8766-37ddff5a7b54-0
01:27:41.800 --> 01:27:44.902
OK, so the thing which you have
to remember is the variables

c9d9d536-b45d-4491-8766-37ddff5a7b54-1
01:27:44.902 --> 01:27:48.005
which you are creating in the
batch step can only be used in

c9d9d536-b45d-4491-8766-37ddff5a7b54-2
01:27:48.005 --> 01:27:49.480
the batch step process phase.

ed9166c7-75c7-4686-befc-fba3d4787b4f-0
01:27:49.480 --> 01:27:52.243
You cannot use them in the batch

aggregator or in the on

ed9166c7-75c7-4686-befc-fba3d4787b4f-1
01:27:52.243 --> 01:27:52.680
complete.

87052adc-30fb-4938-837e-d9fe10019b42-0
01:27:52.880 --> 01:27:56.000
And even the outside you know
you cannot use this variable.

63f79d37-7d4f-4976-b17a-79d32679b971-0
01:27:56.120 --> 01:27:59.076
The variables created in batch
step cannot be used outside the

63f79d37-7d4f-4976-b17a-79d32679b971-1
01:27:59.076 --> 01:27:59.640
batch scope.

297bf1cc-da18-4242-aed6-adb617b506b2-0
01:28:01.560 --> 01:28:02.200
Make sense?

a2671461-c46d-4e7c-b66d-9dedda420fd1-0
01:28:04.520 --> 01:28:06.480
Yes, yes, yes.

2158d678-3545-4315-b67a-fc94da8ac381-0
01:28:08.240 --> 01:28:11.279
So that's about batch processing
and even this is not the entire

2158d678-3545-4315-b67a-fc94da8ac381-1
01:28:11.279 --> 01:28:11.560
thing.

bb4e2f12-4fbe-4cf1-88c5-a64d81cf45e1-0
01:28:11.560 --> 01:28:15.240
So there are again few more
things to optimize them.

e03244ba-e3e9-474e-a40d-fec5a936e9bb-0
01:28:15.320 --> 01:28:19.880
But I think this is what is
enough for the time being.

25f1667f-5019-46a2-9672-2c830dcf78ff-0
01:28:19.880 --> 01:28:20.040
Yeah.

ee5a2220-e3ab-44ff-9d5a-2803c1c58fa0-0
01:28:20.040 --> 01:28:21.080
And any questions?

d42f9d22-22fe-4198-857c-2a37eda34086-0
01:28:27.800 --> 01:28:32.822
Yeah, so 22, it's going to be
last session for the team or

d42f9d22-22fe-4198-857c-2a37eda34086-1
01:28:32.822 --> 01:28:34.440
like for everybody.

be8607fe-8075-4f73-89ce-0dd29811a397-0

01:28:34.440 --> 01:28:39.322
So yeah, this is going to be the
last semi session on 19th, we'll

be8607fe-8075-4f73-89ce-0dd29811a397-1
01:28:39.322 --> 01:28:43.317
be having a session where I'm
going to talk about the

be8607fe-8075-4f73-89ce-0dd29811a397-2
01:28:43.317 --> 01:28:47.460
assessment and the interim
thing, I mean the assignment

be8607fe-8075-4f73-89ce-0dd29811a397-3
01:28:47.460 --> 01:28:48.199
parts, OK.

19c5584b-52ef-4ad4-a4e7-770096dff8c9-0
01:28:48.200 --> 01:28:50.928
So for the topics and semi
sessions, yes this would be the

19c5584b-52ef-4ad4-a4e7-770096dff8c9-1
01:28:50.928 --> 01:28:51.160
last.

ee43795c-dc54-4c3c-b3c3-a8ff7823409b-0
01:28:55.760 --> 01:28:58.400
So Yep, so I think this is the
15th session.

87f12e3d-34c3-44ba-8920-072c1c9764cf-0
01:28:58.400 --> 01:29:00.280
I hope you guys had a good time.

fc3e3796-4f74-446c-895c-9e653834127e-0
01:29:00.280 --> 01:29:03.280
You know, learned few things
from the sessions.

c8b28d8d-1409-4dcb-8b1d-aaed1540086e-0
01:29:03.520 --> 01:29:06.142
So if you have any feedback
regarding any of the sessions,

c8b28d8d-1409-4dcb-8b1d-aaed1540086e-1
01:29:06.142 --> 01:29:08.720
so you can just drop me, you
know drop a mail, that would

c8b28d8d-1409-4dcb-8b1d-aaed1540086e-2
01:29:08.720 --> 01:29:09.520
really be helpful.

af967c1f-a613-4510-9082-154a95fa79e1-0
01:29:09.840 --> 01:29:12.563
And if you have any queries, you
know not only after the

af967c1f-a613-4510-9082-154a95fa79e1-1
01:29:12.563 --> 01:29:15.526
training, also while doing the
job, if you have any questions

af967c1f-a613-4510-9082-154a95fa79e1-2
01:29:15.526 --> 01:29:18.441
you can always reach out on
teams or drop me a mail and I'll

af967c1f-a613-4510-9082-154a95fa79e1-3
01:29:18.441 --> 01:29:20.640
see how I can help you out with
those things.

7c2a8edb-a614-4cd2-be20-20df186480de-0
01:29:22.400 --> 01:29:22.720
Yep.

683a5399-38c5-4b97-b107-3fd93dd02c94-0
01:29:22.840 --> 01:29:24.160
So that's it guys.

80a606fc-0d98-4a73-8f15-59bea07f7487-0
01:29:25.880 --> 01:29:29.898
If you don't have any other
questions, we can close the call

80a606fc-0d98-4a73-8f15-59bea07f7487-1
01:29:29.898 --> 01:29:31.480
and catch up on Tuesday.

da3e6edc-2c41-4fca-876d-97979026f1c5-0
01:29:32.240 --> 01:29:34.520
So we are also doing hands on
like on that.

0d94779d-f2a6-4fa4-842f-714f5bd2e599-0
01:29:34.520 --> 01:29:37.599
So like Siddhartha Will, if you
get any kind of stuck there,

0d94779d-f2a6-4fa4-842f-714f5bd2e599-1
01:29:37.599 --> 01:29:38.760
right, we'll reach you.

2e20528f-2ad3-4a14-b68e-cf605c2bcf42-0
01:29:38.760 --> 01:29:38.880
Yes.

084f561d-b3fd-4f53-bc3d-464e7c5e5157-0
01:29:39.680 --> 01:29:40.920
Yes, please do reach out.

5573d1ab-e9db-44bc-a985-0c8d44508c8b-0
01:29:41.760 --> 01:29:42.080
Yes.

56a6aa3e-3c0f-4d1f-9551-67d0a847c4d2-0
01:29:44.360 --> 01:29:44.640
Yep.

7463223a-3aa7-46df-9ba9-f23f2523545c-0
01:29:45.040 --> 01:29:45.520
Thanks guys.

810ab422-f82d-4aea-bc38-c8c768937e9c-0
01:29:45.520 --> 01:29:46.320
Thanks a lot for your time.

1037c3f6-6fe9-48b7-bf7a-9ce6209c0b4d-0
01:29:46.720 --> 01:29:47.000
Thank you.

2631eb8e-5991-4315-86b3-f8f46053f67c-0
01:29:47.920 --> 01:29:49.560
Thanks for that.

3d65ec36-fe2c-49c7-a63c-5e0c7627c12e-0
01:29:49.720 --> 01:29:50.160
Thank you all.