# Real-Time Pipelines with Flink and Kafka

## Day 1 Lab

## SETUP:

### Transcript:

50:10
credit make sure to switch over to the next tab so that you can get credit to be certified and I'm excited to see you
50:15
in the lab hey welcome to the Flink lab setup so to get started here we're going to want to clone the data engineer
50:21
handbook so you can do that with uh either code here you can do it with SSH or you can download the zip doesn't
50:27
really matter uh make sure that you have it cloned though uh that's going to be the first step and then once you have
50:33
the handbook open you want to you'll see there's a bunch of materials here so you want to do boot camp materials for
50:39
Apache Flink training and then example m.m and in here you'll see it says a
50:45
couple things here like Kafka web traffic get from website Kafka web traffic key get from website so what
50:51
that means is we want to go back to the Flink lab setup uh this link here boot camp. Tec creator. SL lons Flink lab
50:59
setup and then here you'll see there's this your Kafka credentials in this PDF so what we want to do is we want to copy
51:05
uh each one of these over so we're going to copy this guy over into our web traffic secret we're going to paste him
51:12
in there and then we're going to copy this guy over here uh this web traffic key and then we're going to paste him in
51:19
here and um please make sure to not publish any of these keys in any public repo because otherwise confluent will
51:26

get very angry at me so then you'll see we need get this IP to location so the
51:31
IP to location you just go to this website and you'll be able to get uh the IP to location um API key it's very easy
51:39
very clear uh thing to do there I'm not going to do that in this call because like uh I don't want to share my key
51:45
with y'all but make sure to do that as well and then everything else is just set and you don't have to worry about it
51:51
so then what you want to do is you want to um essentially copy this file and paste again and make him Flink
51:58
m.m and we're going to say okay and we should be able to we want to overwrite if you have anyone there so now we have
52:05
this overwritten now this is our Flink m.m this is how we're setting up our Flink environment so the next thing we
52:11
want to do is we want to go to that spot one second let's
52:17
um let's go there real quick let's go to the data engineer handbook boot camp
52:22
materials for Apache Flink training so that's where we're at so let's make this a little bit bigger so in here right
52:30
you'll see uh there's a couple different ways that we can do this so I'm going to do this all with Docker compose and not
52:36
use the make so we have these nice little uh make commands that can make things a little bit easier but what we
52:42
want to do here is we want to do this Docker compose command here uh this is going to set up Docker to run so this is
52:50
going to build our Docker image so we're doing uh Docker compose mile Flink M
52:55
build remove orphans right so then this is this uh the first time for me a lot of this was already built and cached so
53:02
it was faster for you it might be significantly slower because you have to download all the Flink dependencies but
53:08

so what's happening right now is we are doing a pip install for all the rest of our dependencies that we need in order
53:14
for our Flink uh container to get running uh and this takes a little bit
53:22
of time so just just be patient with it and uh we'll be able to get through this together I I Believe in Us uh so all
53:29
this is doing is installing all the PIP libraries that we need to go through so what we're going to be doing today is we
53:34
have we're going to be checking out this thing called start job so start job what it does is it looks at a Kafka topic so
53:41
the Kafka topic that we're going to be listening on is actually the one for data expert so every time you do a page
53:48
load for data expert it puts a record in Kafka right so it puts a record in the
53:53
Kafka and then what we want to do is is uh and you'll see in the Kafka topic the
53:59
Kafka topic name is boot camp events prod and um that's where it's going to be putting its values and uh it should
54:07
be pretty awesome and then we have the start job where you'll see one of the things that has is this IP address so
54:13
what we want to do is this job what it does is it actually takes the data from
54:19
Kafka then what it does is it hits it actually hits a freaking um this API
54:24
location API and it geoc codes the IP address so we can see where the traffic
54:29
is coming from should be pretty cool so uh we're almost done here it's uh probably needs about another like maybe
54:36
10 15 seconds here to uh be fully installed and ready to go but this is uh
54:42
this is really good because this is um like and if you're on Mac you can just say make up you don't have to have all
54:48
of this freaking fancy stuff uh but like the docker compos stuff is the one
54:54

that's more Universal okay so you see we have our job manager and task manager started so what we want to do is we want
55:01
to go to Local Host 8081 and you'll see here is our freaking
55:06
Flink job manager this is where we can see our jobs running and uh there's nothing running right now so uh one of
55:14
the other things that we want to do here oh my bad one of the other things that we want to do here is we need to make sure that our other um from week one
55:22
there is um a postgress container that you need to be have running so make sure you have that running follow the
55:29
instructions on week one to have that running and then what you want to do is you want to go into postgress and then
55:36
you you need to run a small script first so you'll see there's this init SQL so
55:42
you need to take this SQL and then run this SQL okay this SQL just ran uh you
55:48
need to do this in order to get postgress to actually be able to collect the data so now what we want to do is we
55:55
want to go back into our Flink um uh make file so you'll see with this you
56:02
have this make job command and this is going to what's going to kick off the the Flink job command so what we want to
56:07
do is we want to paste that in and run it here Docker compose and um I have it running right here you'll see like it
56:13
ends up kicking off this Docker compose command and then it will create our kofka stream with Flink and then it says
56:20
loading in the postris and then job has been submitted with job ID so you'll know it's running when you go into here
56:27
and you'll see processed events you should see a nice little running thing here at Local Host 8081 if you have
56:33
everything set up correctly there's one final check to make sure you have everything set up correctly and that is
56:39

going to be you want to go and say select star from processed data from processed events and if this oh make
56:48
sure to put in the password there but like you'll notice that this will have
56:54
data and um if there's actual data in here then that is a good sign that you
57:00
did the right thing so yeah uh I hope you enjoy the lab this is going to be really fun we're going to uh obviously
57:07
you want to clone it I sent y'all a a copy of this Flink environment so let's go look at this uh thing here this uh

**LAB 1:**

**Transcript:**

57:14
Flink M so you see this like uh copy example M to Flink M so definitely do
57:20
that there's going to be a um a lot of these values will already be covered and so you don't have to really worry about them um make sure you have uh the week
57:28
one and two database running in postgress otherwise this is not going to
57:33
work and so you want to make sure that that's running um and you don't like also make sure to pull for the latest
57:40
version of this refo because this repo used to be coupled with its own um postgress image but I don't think that
57:46
that's important I think that like just having this be just Flink is the way to go and so then what you want to do is
57:52
after you have that Flink M kind of put here right you'll have see here's Flink
57:58
M with all of the super secret password stuff right cka password right there all
58:03
these different values here make sure you have your IP coding key you got this from the ip2 location website that I
58:10
Linked In the Discord make sure you get your API key and put it here um the code
58:16

will work without this but uh it will be a lot cooler if you put it in and then uh obviously you have like the Kafka
58:22
URLs these are all the Kafka Brokers for my Kafka cluster and then obviously you have like your
58:28
post address your username password all these things like the how to connect the post address it's not going to work without this stuff so you want to make
58:35
sure to have that set up and then after you have that set up you want to go to SQL and then there's going to be this in
58:40
it. SQL and you want to get this processed events table and go ahead and we're going to copy this table over into
58:47
our uh kind of editor here and then we will um we can run this right obviously
58:52
it's going to fail for me or or create table if not exists right so it will work either way if it and then you'll
58:58
have your processed events table so hopefully if you do everything right here um we will see data in this um in
59:07
this table uh after we kind of walk through our Flink job so before we set
59:13
that stuff up and we go into Docker I actually want to walk through the Flink code and then we can go from there
59:20
so in uh in the repo you'll see uh there's a SRC folder SRC job and then
59:27
there's two jobs there's aggregation job and start job so we're going to go into start job which is going to be the one
59:33
that we go with first and what we're going to want to look at here is if you go all the way down there's this thing
59:39
called log processing that's going to be this is essentially like the the start of the job this is where um all of our
59:46
definitions and stuff are going to go from and we can look and see how this job is actually set up so one one of the
59:55
things that I uh I noticed um when I was working through this the very first thing you want to do is you want to get
1:00:03

this stream execution environment so you need to give Flink the ability to say okay we're
1:00:10
doing a stream environment because Flink can also do like uh you can say like enable um batch mode this actually works
1:00:19
right you can actually throw that on or I think it might be is batch mode or something like that this is like you can
1:00:24
definitely Flink actually under understands that and then it will process things a bit differently but the default is streaming so we don't have to
1:00:30
worry about it um then oh yeah then we have enable checkpoint let's change this because this 10 here is actually in
1:00:37
milliseconds so let's go ahead and multiply this by a th so that we do every 10 seconds not every 10
1:00:44
milliseconds because every 10 milliseconds is a lot of checkpointing that's like too much checkpointing and
1:00:50
um then okay then we have oh yeah here it is it's in streaming modee right here that's okay we do but we do set it here
1:00:57
so first we have our execution environment right then we have our settings so these are going to be the settings that we'll uh that we want to
1:01:03
enable which in this case we're just saying okay use streaming then this is what we call our
1:01:09
table environment which is where we can Define our sources which is going to be which this the table environment is what
1:01:16
we're going to use for the rest of the time so these are these higher level
1:01:21
environments we only use at the beginning you can kind of think of them as like the kind of like the spark context and then later on you just work
1:01:28
with like data frames going forward and that's essentially what we're doing here is this T environment allows us to kind
1:01:34
of work more with like uh what the what the kind of the spark equivalent of a
1:01:40
data frame is and then the F the first thing we do is we we do this create
1:01:45
temporary function this is very similar to registering a UDF in um spark if you
1:01:53

know a UDF UDF stands for a user def function and in this case uh we're doing
1:01:59
git location let's go ahead and look at that function and command click and you'll see here what this does is get
1:02:05
location what it does is you have this IP to location uh URL this is from that
1:02:12
website that we were talking about and then what this does is it takes in an IP address and then it takes it uses the
1:02:19
the API key that you uh provided and then it goes ahead and it it tries to
1:02:26
get the country region and city of the of the data provided and you'll see that
1:02:33
like um if you don't give it the key the request will fail and if the request fails it will just return an empty
1:02:39
dictionary so if you don't want to make an account it's fine and we'll still get data it just won't be as cool um and so
1:02:45
that's what this UDF does is it takes in an IP address and it returns a Json object of country state and city that's
1:02:53
what we're going to do with uh this uh UDF function and you'll see it's called a scalar function the reason why it's a
1:03:00
scalar function is scalar functions take in one row or one column and they return
1:03:06
one row or one column so in this case we're taking in one column IP address and we're returning one column this Json
1:03:14
object so that's what scaler means there's other ones like aggregating function and these other types of udfs
1:03:20
as well that you can work with but in our case we are just using scalar function and if you're not familiar with
1:03:26
python this is actually how you do inheritance in Python so our class is git location but it it inherits the
1:03:32
scalar function uh functionality and then all we need to do is implement this and that allows us to use all the other
1:03:39

functions in Flink that are associated with scaler function with this without having to actually Implement them

1:03:46

ourselves okay cool that's the git location UDF this is what we're going to use to enrich our data with or enrich

1:03:54

our event data with location data based on IP address so if we go back down here

1:04:00

let's go back to that git location call right so this is where we're registering git location great so then what we want to

1:04:08

do is we're going to create a couple different uh tables so we're going to create a source table and then we're

1:04:15

going to create two syncs we're going to create a postgress sync and a cka syn

1:04:20

and our source table is actually a Kafka source so you see here we say create a

1:04:26

Source Kafka so if we go into here this is where uh we actually go and Define

1:04:32

our table and you'll see okay our table name is events and this is something

1:04:38

that is kind of we have to call it events because that's what I called it uh when I set everything up and then uh

1:04:44

you'll see in here we have our um let we try to say here this

1:04:50

is okay so then we have um a pattern so I need this pattern to like essentially parse the time stamp cuz uh in Kafka the

1:04:58

event time is actually a string but we want to turn it into a time stamp and this is the pattern that the time stamp

1:05:04

needs and then what we do is you'll see here you see this Watermark uh code this

1:05:09

is where it diff this is like you'll see that this like initially you're like wow this is squel and then you're like wait

1:05:16

a minute this isn't squel what the hell is going on with this four keyword right this is like a real really weird one so

1:05:23

what this is saying is the Watermark essentially this is giving it that 15c

1:05:28

window I was talking about right and that like so this will fix the ordering of events that are within a 15-second
1:05:35
window and so that is uh what this Watermark is doing and then this part is
1:05:42
so nasty this took me so long to figure out how to do but anyways what is going
1:05:47
on with this let's talk about each step each one of these configurations cuz uh
1:05:53
I don't want to just gloss over this and be like yeah this is how you connect to Kafka so um so you know how someone was
1:05:59
saying at the beginning of this lecture or the beginning of this lab are Kafka and Flink connected they are not and
1:06:05
you'll see this because you see the connector here is kfka and if say we were using rabbit you could also do
1:06:11
rabbit mq here if you were using a rabbit connector or this can also be
1:06:16
jdbc which is uh the Java database connector so you can actually have Flink also read from postgress or from another
1:06:23
relational database so so like Flink can pretty much read from wherever the hell you want it to read from it's not that
1:06:30
big of a deal and then you have the bootstrap servers so these are going to be all of the the the servers that are
1:06:37
running Kafka this is usually like a cluster of servers that runs Kafka together and then you have the topic so
1:06:44
you can think of a kofka topic is very similar to like a database table and um
1:06:49
if you look in the Flink environment in this case our Kafka topic is boot camp events well technically it's porcupine
1:06:57
78 436 boot camp events but uh I I couldn't get rid of the porcupine stuff
1:07:03
cuz uh so I use Heroku Heroku kofka that's how I set up kofka because I tried to set it up on AWS and do it all
1:07:10
myself but then I was like wow this is a lot of work so I just decided to use heroku's Kafka which worked a lot easier

1:07:16
and so then we have the Kafka topic which you can think of a Kafka topic and a database table are kind of similar
1:07:22
then you have uh this SL SSL endpoint I identification algorithm this can be
1:07:27
https and other things as well but um Heroku uses uh um key stores so it uses
1:07:34
these four to do the algorithm to identify that you are someone who has authorized access so we just have to
1:07:41
essentially say this is nothing because there's like this weird bug if that happens so then you have the Kafka group
1:07:47
ID so Kafka can have a group so you can have um you can think of this as like a schema so you know how like a lot of
1:07:53
times in in the database world you have like a pr. name or dev. name or uh you
1:08:00
know the different uh it's kind of like that higher level grouping so this can be that's what this is in kofka and
1:08:06
you'll see that that's in this case this is called a porcupine 78 436 boot camp
1:08:12
so that's going to be our uh kfka group that's what we're going to be using for that and then we're connecting over SSL
1:08:19
so this is actually secure Kafka and then uh these ones get kind of gnarly
1:08:25
this is essentially the password and uh the the the different trust certificates
1:08:31
if you want to look at these files they're actually in this uh you'll see there's this folder called keys and you'll see it's like super
1:08:38
nasty and encrypted files right they're both like really crazy this encrypted files that you
1:08:44
could decrypt with the kofka password and then that gives you access to the Kafka um database or to the Kafka you
1:08:53
know cluster I mean not database then after that we have a couple other things here these are the last three and then
1:09:00
we have gone over the the all the million things that you need to connect
1:09:05

to kofka so then uh remember how I was talking earlier about how there's latest offset and earliest offset so what I'm
1:09:11
going to do here is I'm just going to sneak in here and I'm actually going to change this to uh we're going to change this to latest offset and then this can
1:09:17
also be latest here so what this one does uh scan startup mode so this is
1:09:22
going to be when you first kick off your Flink job is this going to read from the first record in Kafka or the last record
1:09:29
in Kafka right and then this uh property's Auto offset reset this is if
1:09:34
it fails and restarts do we uh read from the the first offset or the last offset
1:09:40
and that's what uh this is going to do or it can also pick up from a checkpoint so that's a a separate thing so it but
1:09:46
checkpointing is optional so this is but this is required so um even if checkpointing ends up failing then it
1:09:53
will this will end up picking up where it left off the last thing is like okay how is this data stored and then in
1:09:59
Kafka this is stored as Json you can also put like CSV and tsv and other
1:10:05
values in here as well but um the data here is stored in Json and that's how my
1:10:10
Interceptor uh dumps the data to Kafka so that's why we have to use Json here so this essentially gives us a schema
1:10:17
that we can work with and we can play around with so then what we want to do here is we want to execute the the table
1:10:24
environment so the the table environment is saying okay this table exists and this is how we connect to it so then
1:10:30
this gives us access to this Kafka que great that's how watermarking works
1:10:35
and this number could be bigger or smaller I chose 15 seconds and it's probably it is a little bit arbitrary you can play around with it to kind of
1:10:42
understand like how ordering works but yeah watermarking is is is important for out of ordered events though

1:10:48
so yeah yeah no problem thanks that was a really great question so um so we've
1:10:54
defined our source which is costum now let's look at uh let's look at our Sync here which is
1:11:01
we're going to have create processed events sync postgress in this case right
1:11:06
um we uh we actually don't necessarily need this Watermark here because we we you only really need it in one spot but
1:11:12
in this case we're making another uh table here so in this case what we're doing is we have a table that has a
1:11:20
different schema right you'll see it has this new Geo dat column which is also varar
1:11:26
but what this does is you'll see the connector here is different in this case we are connecting via
1:11:33
jdbc our URL is our postgress URL that's defined in that Flink environment and
1:11:38
then we say our table name table name in this case is processed events and then we have our username and password and
1:11:44
then we have the driver so this is going to be our driver code and this is something like when you're deploying
1:11:50
Flink you have to include this uh this Java code in your deployment if you want to be able to connect to postgress you
1:11:57
have to include the driver code as well otherwise this doesn't work so that is essentially how uh the sync works for
1:12:05
postgress this allows Flink to directly access this postgress table processed
1:12:10
events and keeping in mind though that this create table statement does not actually create a table in postgress
1:12:18
this is just to have Flink be aware of the schema so that's why you actually have to run that you know how I said you
1:12:24
had to create this create table at the beginning of the lab you have to run this create table it's because you
1:12:29
they're different right and so uh this is just to let Flink know what the schema looks like um so then you have uh

1:12:37
so that's our postgress sync let's look at the last sync and then we can talk about how uh each of these things
1:12:43
actually works so then we have the Kafka sync so here's a Kafka sync we have process events kafa and then you'll see
1:12:50
in this case we don't use this pattern because we don't need to because it's a sync um so then we have uh we have our
1:12:57
um sync ddl which is all this is the same you'll see this ddl and the postgress one are the same because these
1:13:04
are literally the same I I just wanted to show people how you can write data both the postgress and the Kafka um and
1:13:09
you'll see in this case all of this is exactly the same as um in the source
1:13:15
data because it's the same cluster you'll see the only difference here is actually the topic you'll see the topic
1:13:21
here is a little bit different it it's funky how it works right now like I'm doing it this way where I do this like
1:13:26
weird split logic so that we can get that weird porcupine uh thing out of it that like Heroku gives me but then I
1:13:33
give it the new table name keeping in mind that like uh like and I'm sad about this because this was something I
1:13:39
thought heroki would let you do but you can't change this name and create a new topic Heroku does not let you create new
1:13:46
topics on the Fly and then in here you then can create your topic name and then
1:13:51
add it to this big old thing and then this is then our sync and this this is where we can write data out and so we
1:13:58
now have our sync and we have our two syns and we have our um source so we
1:14:05
have now gotten through the first five lines of code of this freaking Flink job
1:14:10
and now it's uh let's kind of like actually go to where some of this transformation logic is happening okay so this gives us our

1:14:18
Flink postgress and a source table and you'll see that like one of the things that's really cool is this actually
1:14:23
you'll see this um just returns table name so after you create this table you
1:14:29
can actually reference refer to this table just by its table name just like
1:14:34
you would in a normal SQL environment and I like Flink SQL for that reason and that's one of the things that's really
1:14:39
cool about Flink SQL because Flink SQL and uh Flink Java are going to have like
1:14:45
the same performance um and then in this case what we want to do is we're saying okay we're going to add to the kfka sync
1:14:51
insert into and then this is what we're going to do is we're going to to look at
1:14:56
our IP address right and we pull the IP address this way because it's like kind of in a nested field cuz Heroku actually
1:15:04
does this weird like IP address forwarding thing that you need to uh kind of extract it out and then we want
1:15:10
to do a a date field here um so this is essentially moving this back into like a normal time stamp field because uh it's
1:15:18
it was having a hard time serialize kfco was having a hard time serializing this to Json because it was like a Time stamp
1:15:25
column and time stamp has a harder time like serializing to string but this serializes it back to string and then we
1:15:32
have the referr which is like in terms of web traffic this is like where they came from like LinkedIn or Twitter or
1:15:38
wherever right and then you have the host in this case this is like Zack wilson. Tech or exactly.com or the all
1:15:45
there's like there's like a bunch of them and then you have the URL which is like back SL signup or back SL login or
1:15:51
back SL Boot Camp or whatever and then you have uh this is you'll see this is
1:15:56

actually the git location call so this is actually calling that UDF that we can
1:16:02
then uh call you see how we we we created the git location up here and
1:16:07
then we are calling it in um SQL right here and then we are reading from kofka
1:16:13
that's essentially what source table here is and then this is inserting into the kofka sync which is the new topic so
1:16:20
we're moving we're we're enriching it with data and then we're moving it
1:16:25
so then we have uh the second one here it's doing literally the same thing
1:16:31
where we are taking it and we are uh instead moving it into postgress and
1:16:37
this is going to write records out to postgress and you want to add a weight here on the last line of your code uh in
1:16:44
Flink otherwise this will just uh will run uh for every record that's in the
1:16:49
current kofka q and then it will end the job and this this makes it run continuously this dot
1:16:56
weight so um then we have a little try catch here to in case there's a failure
1:17:02
for postgress and uh this could also be a failure for Kafka but like it's Kafka is probably not going to fail because
1:17:08
it's like Kafka to Kafka which is going to be like less of an less of a bug so then at the bottom here obviously you
1:17:14
have the actual um like the entry point for the job where we actually call the
1:17:19
job so this is the job that we're going to be working with today to actually
1:17:24
kind of go over what's going on here to show like hey look we're we're doing stuff so then what you want to do now
1:17:32
that we have uh that working we can actually go into our messy Docker environment now so what we want to do
1:17:39
here is we want to say make down I'm going to do make down first which is going to we want to uh kill all of the
1:17:46
um the the images that we have up then we say make up

1:17:55
okay so now our job or our um our environment is up and running so let me
1:18:01
show you that environment real quick so if we go to Local Host 8081 you'll see okay so this is our task
1:18:08
manager and you'll see uh we have no jobs so this is um and here's our task
1:18:14
manager this is where it's running and you'll see all sorts of different things about like Flink the memory you can see
1:18:21
all sorts of crazy stuff but the key the key thing here is you see there's no jobs cuz I just started up the task
1:18:27
manager and let's kind of go over like what's going on here with our slots right so we have three available slots
1:18:34
so that means I can run three jobs at once but if I have more than three then uh like it's going to wait for one of
1:18:40
them to finish or something like that but our jobs are actually designed to never finish they're supposed to run 24/7 because they're streaming jobs but
1:18:47
um Flink jobs can also be batch jobs that's why they can finish that's why some of these have finished right so
1:18:53
flink's kind of a one where it can do both so then um what I want to do here
1:18:58
say make job so this is going to call up uh the docker container again and it's going to
1:19:04
see how it's saying okay Docker compos exact we're calling the job manager and we're like okay we're calling Flink
1:19:10
directly and then we're calling the the start job file and then uh you'll see
1:19:16
okay now we have two jobs that have been submitted these are the jobs to write to Kafka and the job to write to
1:19:22
postgress so f link is similar to spark in this way where you know how spark
1:19:29
they say spark is like lazily evaluated and it only actually fires up a job when
1:19:35
uh like there's like an insert statement or a collect statement or there's like a movement of data Flink is very similar

1:19:42
like that and that's why there's two jobs that end up getting fired here and that's because of these insert statements here these insert statements
1:19:48
caused two jobs to be fired and uh that's what's going on um if yall like are curious about like why there's two
1:19:55
jobs instead of like one big job so you'll see here now we have uh we have two jobs here that are running and if we
1:20:02
let's let's go into this processed advaned so if we look at this guy this is going to be our um our code here and
1:20:10
you'll see it's it's going pretty well uh back pressure zero busy zero because
1:20:15
if you remember I changed the source to be uh you see how I changed the source to be latest offset so it's only it's
1:20:23
only reading in new data so like let's let's let's like load up a row okay so
1:20:29
that's going to cause it to have at least one row of data that um let me go into here real quick I need to let me
1:20:36
delete the data here first so that we can see the new data so okay so I
1:20:41
deleted the data from uh from processed events and let's uh add one more record
1:20:47
to the que and then what we want to do here is if we go back here we say select
1:20:52
star from processed events okay so you see I have one record here
1:20:58
and see here is US California City San Francisco right so and if any of y'all
1:21:04
freaking uh want to like go to Zack wilson. Tech you'll see that your data and location will show up here and um
1:21:11
and obviously I live in San Francisco so this is actually pretty good and kind of creepy I I was able to do this and um
1:21:17
and so the idea here is if you see I can go to other Pages too so if I go to like boot camp right oh let me go
1:21:25
exactly boot camp It's like because I'm like logged in over here okay there we go so now you'll see I I should have

1:21:31
generated a bunch of events there so if I like put a limit let's put limit 10 here so there we go oh and some of y'all
1:21:38
were like refreshing stuff too so you'll see oh look at that look A Bruno I think this is Bruno I guess this is Bruno here
1:21:44
Brazil what's up everyone else do we got representation here oh dude we got a lot
1:21:50
in here we got some Florida got some Alaska look at that dude I got some freaking people in here here's got some
1:21:55
Texas right so um Sugarland Texas interesting it's a cool place so anyways
1:22:03
you'll see uh like but you see how all this data was just available immediately
1:22:08
because that's what real time's all about is it's immediately available that's what it's all about and you'll
1:22:14
see um this guy here actually um you can see some of these people are actually coming from different spots like this
1:22:20
guy came from my substack so this is actually probably an event that was not from the boot camp because you see like
1:22:25
there's a referrer here that is um my substack so they came from my substack newsletter and um you can see oh that's
1:22:33
kind of like how all this is working right and this is the idea behind uh how
1:22:39
this works but like what you can do right is you'll see with the Geo data so if we like Geo dat and we Json and then
1:22:45
we say uh country right then we can like um there we go and then we can also
1:22:53
then uh uh count one and we can say uh what group by one there we go then you can see all the
1:23:01
different uh countries so we got us as uh leading we got Nigeria we got Canada and we got Brazil right we got we got a
1:23:08
couple of people showing up from all over the place here right and uh this is essentially what's going on but it's in
1:23:14

real time right so like as more data comes in see there's another one see now we have another one ve prob like
1:23:19
Venezuela I have no idea um uh but like you'll see like this is live data feed
1:23:25
right of this processed events because my Flink job is just processing new data as it comes in and so this is pretty
1:23:32
cool um so one of the other things this job is also doing right because there's two jobs here right so if we saw here so
1:23:40
you see processed events kofka you see it was here at this like 39.67 megabytes and you see how like now
1:23:47
it's a little bit higher it got moved up to this 40 megabytes and now it's like slowly growing and like so that's what's
1:23:54
going on with this process events Kafka because we are writing more data to this Kafka queue as it comes in as like more
1:24:01
and more people like you know spam my website and stuff like that and so this is kind of the idea behind how uh Flink
1:24:08
works is you have um kind of different values that can come into place here so
1:24:15
one of the things I want to show real quick though is I'm going to kill these jobs real quick so I'm just going to I'm
1:24:21
going to say uh so the jobs are killed and I'm going to say make down I'm just going to kill the whole I'm going to start over here and what I want to do
1:24:27
here is I'm going to change this to earliest just to kind of show you the difference right so you'll see here when
1:24:34
we ran that what there was like see how there's like I don't know like 50
1:24:39
records because that's like how many people hit the website in the last like five minutes but now when I run it this
1:24:45
time with earliest offset if we go look at the the the Kafka dashboard so ours
1:24:51
is boot camp events so this is going to be our um actual data set that we are
1:24:56
working with right so this data set is like about 15 megabytes so I would hope

1:25:03
like if we use earliest offset we're going to process all of this data again it's going to process all the way from
1:25:09
the beginning and that's like what we're going to do when we redeploy this job here so in this case we're going to say
1:25:14
make up okay so now what we want to do is we
1:25:20
want to then say make job and then then I'm going to show this is going to create some interesting
1:25:27
things in the UI that I want to also uh illustrate to y'all so that you can
1:25:32
understand some other kind of pieces of this puzzle so there we go we have our two jobs running so now let's go back into the Flink UI while like we can
1:25:39
still uh show what's going on here so let's go look at this uh process events job uh process events job and you'll see
1:25:46
one of the things you'll see is like you see how it's like red right now and it says it's very busy a 60 64% red and so
1:25:54
what's going on here is um it's busy because of the fact that it's oh there we go now it's really red right so now
1:26:00
it's like really busy now it's like essentially maxed out so what happens here is you have um you see how there's
1:26:06
like busy and then there's back pressured so these are two different kind of things that can happen because
1:26:12
so busy means okay we are at Max compute like this this Flink job cannot process
1:26:18
any more data and um if it processes more data it gets moved into back
1:26:24
pressure and essentially back pressure is the second layer where back pressure essentially says it tells Kafka Slow
1:26:30
Down slow down because we we we need to um essentially add a little bit more latency to the data so that we don't out
1:26:37
of memory and that so that we can process all the data that's coming in and this can happen like you know if
1:26:43

your uh data set goes very viral or if your data set is like um like if you get
1:26:49
a lot of hits it's very spiked you get some spiky data sets because you can't necess necessarily manage that um
1:26:55
yourself because you can't just be like Oh I'm going to stand up more Flink machines obviously you can set the parallelism and have like do distributed
1:27:03
compute with Flink as well but you have to pick the number of machines ahead of time and you don't know how much data is
1:27:10
going to come in you could have more could have less right so that's essentially what's going on here right
1:27:15
is now we have our data and it's um essentially running right you'll see uh
1:27:20
now I bet it's back to okay so yeah it's still processing stuff it's still dumping everything out so um this is
1:27:27
kind of the idea behind how this stuff works is we have our um back pressure
1:27:33
and uh our uh busy busyness of our job and you'll see the Kafka one is the same
1:27:39
way right it's just maxed out like trying to just trying to struggle through it and then eventually it will
1:27:46
uh it will like be essentially go from like 100% to 0% because it's just going
1:27:51
to finish everything and uh it will have processed everything but now if we look at like this query here right so
1:27:58
remember like we had like 50 rows here right I bet there is a lot more than 50 now right oh yeah look at that look at
1:28:05
that there's a lot more than 50 now so now we have like uh okay see we have like we have our us we have India Great
1:28:11
Britain Canada whatever IE is uh Japan right all these different countries here
1:28:18
uh even some that didn't even uh you have null here so that means that some of these uh IP addresses didn't Geo code
1:28:25
correctly or they like they have like a there's like a null right but you'll see like I don't know there's 44 different
1:28:31

countries so people in 44 different countries have visited my website in the last day and that doesn't I mean it
1:28:37
might not even be done right let's uh we put like a like a put
1:28:42
a it should um keep processing there we go so you see how it's still it's still
1:28:47
struggling through right all of the data here and that's one of the things that is tricky about postgress sinks so kopka
1:28:55
syns are generally a little bit more performant because of the fact that they um are meant for high throughput because
1:29:03
you have like the uh the the queue can just be dumped in and it it's fine whereas postgress is not right postgress
1:29:10
is more meant for uh like transactional data that is like kind of not large
1:29:15
volume right it has like a and so you want to be aware like when you're building out your Flink jobs like where
1:29:22
is the data going and like what this is wild this is not supposed to be so busy
1:29:28
why is this so busy all the time like y'all need to like quit visiting my website like last time oh that just shows you like because I did this same
1:29:35
presentation back in May and like it was busy for like it was busy for like a minute and now we're like we're rocking
1:29:42
almost four minutes now guys like so this is this is madness but like this is the idea behind like how um you kind of
1:29:49
build Flink jobs but the key thing I was trying to say here is back to the sinks right so Kafka is going to be a very
1:29:55
high throughput sync that like can manage things a lot more effectively so I imagine that the the Kafka dump here
1:30:02
is going to end up being not busy a lot sooner than the postgress one just because of the throughput of kofka and
1:30:09
then uh that's kind of the idea behind how these jobs work congrats on getting to the end of streaming data pipelines
1:30:16

lab Day One streaming pipelines are crazy right if you're taking this class for credit make sure to switch over to the next tab so you can get a credit