EPISODE 1432

[INTRODUCTION]

[00:00:00] KP: The data lake architecture has become broadly adopted in a relatively short period of time. In a nutshell, that means data in its raw format stored in Cloud Object Storage. Modern software and data engineers have no shortage of options for accessing their data lake. But that list of options shrinks rather quickly if you care about features like transactions. Apache Hudi is a platform for building streaming data lakes that is optimized for large engines and batch processing. In this episode, I interview Vinoth Chandar, creator of the Hudi project and founder and CEO at Onehouse.

[INTERVIEW]

[00:00:39] KP: Vinoth, welcome to Software Engineering Daily.

[00:00:41] VC: Hey, Kyle. Great to be back again on the show. Thanks for having me.

[00:00:45] KP: Well, before we get into our main topic, I'd love to refresh listeners with a little bit about your engineering background. Can you tell me where you got started?

[00:00:53] VC: Yeah. So, chronologically, getting out of grad school, I started my career on the Oracle data replication team, worked on the Oracle database server, data replication products like Streams, Golden Gate, and got exposed to a lot of stream processing CDC there. Then my next role was as the engineering lead at LinkedIn on this key-value store back in the day called Walmart, which this was like a Cassandra, like Dynamo based key-value store that we built on scale and supported LinkedIn through all the hyper growth phases. Then, I went on to work at Uber for almost five years, working on various aspects of Uber's infrastructure, data infrastructure, and that's where Apache Hudi happened, and we created a project. And right after that, most recently, before, currently, I'm the founder, CEO of Onehouse. But before that, I was principal and shared confluent. I was working on Case SQL, Connect, Kafka Storage, and a bunch of things around stream processing in general.

[00:02:01] KP: And what does Onehouse do?

[00:02:02] VC: Yeah, Onehouse wanted to bring manageability, like fully managed data lakes to life. We, having supported the Hudi community for almost five years now, while we have great technology, we often find that companies still take a long time to build a team and kind of build, integrate and operationalize data lakes. So, with Onehouse, we are hoping that this easier part for companies to stand up their data lakes and get started on all the open data infrastructure stack out there.

[00:02:37] KP: And for listeners who don't yet know about Hudi, can you give a quick summary?

[00:02:40] VC: Yeah, absolutely. So, Hudi, at its core, adds transactions, updates, deletes, on chain streams on top of Cloud Storage, or HDFS or Hadoop Compatible Storage, if you will. So, you can think of Hudi as a database layer that sits on top of your cloud storage, and providing these functionalities. So over time, though, the project had a rich set of contributors and we've been able to build out a lot of platform components within the project as well. So right now, you just don't get like a raw library to do updates and transactions, but you also get a streaming ingest service. We are working on a caching service. There are a lot more platform components that Hudi provides that helps you build your data lakes very easy with the underlying technology.

[00:03:36] KP: Does it make sense to compare and contrast Hudi with Snowflake or DataBricks or solutions like that?

[00:03:43] VC: Yes and no. So, if you compare Hudi with Snowflake, Snowflake is a managed cloud data warehouse. That's at least how I look at it. And it has some of these core technologies that you can find in Hudi. But it's all like offered to you in a fully managed product, right? And DataBricks, DataBricks itself, the core product, I think of it as a data science workbench, data engineering, Spark clusters as a service. And while Hudi does not provide a query engine, Hudi just simply interoperates with every query engine including like DataBricks and even Snowflake down the line. Databrick does have delta lake, for example, if you heard about it, which is more comparable to some of the core components that you will find in Hudi.

[00:04:32] KP: And I know Hudi runs well on top of HDFS. Does it run in other settings as well? I know like S3 buckets or blob storage is sometimes used for data lakes.

[00:04:41] VC: Yeah, so that's a popular misconception. Hudi has been supported, for example, on top of Cloud providers officially been supporting Hudi on Cloud Storage for two years now including AWS. So, Hudi can work with any Hadoop compatible storage layer that includes all the cloud stored.

[00:05:02] KP: And you'd mentioned Hudi, if I understood correctly, doesn't have a native querying engine, but has interoperability. Can you expand on how I get access to my data?

[00:05:10] VC: Yeah, so let's try to raise a part of like data engineer who has some data in a Postgres database, and then wants to query the data on the other end. So, what you typically do is you stand up the [**inaudible 00:05:24]** that's outside a Hudi. And once you have that, let's say, your change logs, and Hudi gives you tools to simply run a Spark command. And then it kicks off a Spark job, which can now store and write build a table in some S3 bucket for you or a Cloud Storage bucket for you. And then Hudi registers the table into a meta store, like the hive meta store, or like if you're an AWS, AWS Glue. And from there on, you can use, manage the Presto offerings, like AWS Athena, or run your own presto, or Trino. All of them can query on Spark, of course. You can query it as just another Hive table, if you will. And Hudi also has very optimized reports were for, let's say, for Spark and for Spark, you can query it as a native Spark data source. And you can even obtain CDC chain streams from your tables using some of these more native engine integrations.

[00:06:31] KP: And what's the onboarding? If this is the right tool for an organization, what does it take to get up and running?

[00:06:37] VC: Yeah, if this is the right tool, that's where I think we made it, like very simple. We have two ways to get your data in. If you're coming from the Apache Kafka streaming world, you have a lot of streaming data, there is a Hudi Kafka Connect sync that you can use to start building your Hudi tables on top of Cloud Storage. Or if you're coming from the more, the data engineering Spark world, then you can start with this streamer tool that we have just to get your data into your data lake.

So, those are very simple built-in platform components and maintained right in the Hudi project. And these tools also support a variety of data sources already, like it can pick up any data dropped into a Cloud Storage bucket very easily, Kafka, other streaming sources. And once you get your data in, then you can pick up any query engine, write pipelines, or query the data build dashboards. And even if you have existing tables, you could move to Hudi to get faster runtimes for these pipelines, make them more incremental using the upgrade capabilities that Hudi has.

So, it's as simple as in writing and reading from any other table that you have. Hudi hides all of the underlying complexity around transactional snapshotting, and these kinds of things under the hoods for you.

[00:08:02] KP: Well, let's imagine a startup that maybe released an app, like a weekend project. They unexpectedly got a lot of signups and this thing takes on a life of its own. They're probably not thinking about the BI infrastructure out of the gate, but maybe they're doing a good job saving all their data, and there comes a point when they hit some sort of critical mass, and they really need to take some steps to be able access that data. Is there a checklist or set of indicators or heuristics I can use to decide if Hudi is right for me?

[00:08:30] VC: Yeah, definitely. I think in this scenario, so this is a deeper question, right? So, this data, let's assume this data is now sitting in some cloud storage as some JSON files or something like that, something more unstructured. So now you have different paths for you to take. Broadly speaking, you could send it to like a browsing stack, like a cloud warehouse, like Redshift, or Snowflake, or BigQuery. And if you just want to build some dashboards and reporting on top of it, you probably have like managed solutions there to kind of like auto loader or something to get the data in, right?

Similarly, though, if you want to build, however, towards a model, where you want to eventually run data science workloads on the same data, I think that's when the Hudi path is really good. You can pick up our streaming tool, you can bootstrap all the existing data into a table, and then you can even start continuously ingesting new signups or whatever that's landing in the same location, and then start building out these tables on top of, like backed by open formats like Apache Parquet, which is what Hudi uses underneath and explicit across everything.

So, then you'd be able to do the same thing, use Presto, let's say for your dashboarding while you started running Spark jobs, and maybe your setup for data science down the road, as well.

[00:09:54] KP: So, one option you'd mentioned, I guess I could have parquet files to start with that have the schema kind of embedded in them if they're created right. But something like a JSON or JSON L file, it sort of maybe implied. I don't want to call it unstructured data, but it's semi structured. How does Hudi or does Hudi help me in any way in inferencing what the schema is?

[00:10:14] VC: Yeah, so right now, not any more than what is the Spark JSON source will help you. Hudi doesn't get anything more on top. But we built some of that intelligence before at Uber, and we will allow to down the line, with also Onehouse, contribute more useful things like that, which will schematize the data. I think you have a great point there where I think, if we can schematize this data, I think we can also do a lot more data quality enforcement at the entry point and the lake, which is a huge problem for folks in terms of building trust on the data that you ingest there.

[00:10:52] KP: Yeah, absolutely. Well, you'd mentioned that Hudi can offer some transactional services that I could make transactional updates to my data lake. Could you expand on that? Because it's a distributed system, so there's still the CAP theorem sitting there.

[00:11:07] VC: Yeah, so architecturally, if you think about it, Hudi has a very different architecture, right? If you contrast it with, let's say, even like Walmart, where Cassandra, if you will, if you want a more popular example, there, the cap you had individual storage nodes, storing the data, and then how you read and write, how you configure quorums is how we capped it and played out for you. Here, actually, the data is all sitting in S3. So S3 actually deals with the cap theorem aspects of it, all Hudi brings is a horizontally scalable compute layer, if you will, that is performing these updates and/or deletes. And then also, tracks metadata right on top of S3, again, or any cloud storage interchangeably, where you can – it's pretty much like as consistent and as available the partition tolerant as the underlying Cloud Storage is. So slightly very different model in terms of other distributed data systems that I've worked on, including Kafka or like other distributed databases in general.

[00:12:16] KP: And do most people stand up their own Hudi or their commercial Hudi as a service offering available

[00:12:21] VC: Hudi, so like I mentioned, I think five cloud providers have been offering commercial support for Hudi for over a couple of years now. And the value add is for customers is let's say, you pick up Hurry along with their other offerings like for EMR, the EMR team makes sure the latest Hudi version that they offer on EMR runtime and all of that work well together, right? That's what to the extent that Hudi has been commercialized. And we recently announced Onehouse, earlier this month. And our goal there is again, not to commercialize Hudi. We don't plan to have an enterprise fork for Hudi, but we are trying to build as a service part, which is like typical use cases that people do on top of Hudi, can we offer a managed service that lets them do that. That's what we are focusing on, while we understand that large part of the community can take Hudi assets and run it by themselves. Because there is like all of these other great platform tools already available in Hudi.

[00:13:30] KP: Could you expand on some of the use cases that you've seen being popular in adoption?

[00:13:35] VC: Yeah, so the first one I've seen is just like database CDC, going back to your example of that small startup. Startup has a lot of signup data in Postgres. Now, they've dumped it to JSON files on Cloud Storage. But if you want to get like an equal end table on the lake, that theirs is, let’s the JSON has changed records, then you need a system like Hudi to be able to take these database change logs, and then apply them to a table incrementally with good performance. And that's where, a lot of people have an update for Hudi.

This is very similar to the EL model that you see on the warehousing, cloud warehousing stack where people replicate the databases right into like a Snowflake or a BigQuery, or Redshift. That use case is like extremely popular, I would say, for Hudi. I think there's a reason Robin Hood blog that came out where they talk about how they're able to do like minute level data freshness for that kind of like use cases.

So, the second is, I would say a subset of that is getting that done very quickly, in like near real time. The second large use case would be further down the line. Not just ingestion, but when you're like writing EDLs, a lot of people use Hudi as a way to merge. So, if you look at contrasted warehouses, they've had merge statements for a long time. And for lake, I think this is pretty new. It's recent in the last couple of years. So, when people are migrating EDLs off from warehouses onto the lake for various reasons, Hudi provides the SQL DML support, like your SQL DML statements, so that you can write these in a more incremental style.

And the third large use case I've seen is just for GDPR, Hudi comes with also a lot of indexing on top of your tables, and this was actually one of the use cases that I believe pushed the category itself mainstream, which is now with the GDPR, and like more and more privacy data, privacy laws being enacted, we can't afford to treat lake like a dumping ground anymore, right? You have to actually property manage it. We need to know if a user leaves your service, you need to be able to go and delete the records of that user. So, it provides a lot of like indexing capabilities that help you do also, deletes on top of this data. So, this is also another use case that I've seen Hudi commonly used for to enforce this kind of like compliance activities.

[00:16:16] KP: We've touched a couple times on Hudi being a good option for data science workflows. I always think of those as a little bit more challenging than your typical BI report, if an executive wants sales by region, by day, as long as you get your query right, the answer is just there in the analytics. But when you're trying to maybe build a training dataset, you need to do feature engineering and detect sparse values and all types of things like that. Can you talk about the user experience for data science workflows?

[00:16:46] VC: Yeah, so I think this goes, I would probably start one level higher. I think this is where even architecturally, like the legs are really, really cost effective for data science workloads. As you rightly pointed out, the problem for analytics is more about, I have a large amount of data, how can I quickly like sell their dashboard by pinpointing or like reducing the amount of data that you scan? For data science, for future engineering, the problem is kind of opposite. It's a throughput problem. How can you efficiently cost effectively scan large amounts of data, right? And then sift through it in a cost-effective way. I feel like most of what Hudi offers us benefits is what you generally get with the architecture on the data lake, where data just sits

in a cloud data storage, and you have horizontally scalable data processing frameworks like Spark or Flink that can go sift through the data.

Beyond this, there are certain things that I think we believe, improves the efficiency or improves the lives of data scientists. One is, again, the ability to even pick a data that is going on and being able to update your feature stores. You've seen people build feature stores, and these features have to be kept up to date, as on a daily or hourly or even like, in some cases at Uber, we we update features, even in real time and near real time at different cadence. So, that's another place. But the core capabilities for updating, and also time travel, and having like different versions of the table expose to data scientist, you're able to now run your models on different versions of the table, cross compare. These kinds of additional capabilities, I believe, are very useful for data scientist, from what we've seen, at least in the community.

[00:18:45] KP: And Hudi is a relatively new tool, but has seen some strong adoption. You've named a couple of the companies that have picked it up and are using it, feel free to drop a few more if you want. But it's a tool being taken seriously by large companies. Why is that?

[00:18:58] VC: Yeah, it's a great question. So, I think a lot of these companies have either come to a point where they feel like, “Oh, we want more open data, you know, format to manage all of our data. And in a more like query engine, vendor neutral way, while we retain all the manageability that comes with it, like, you know, the indexing. In open source, we have some of the best data optimization techniques like Hilbert curves that you find in warehouses, or clustering, data clustering zero curves. So, all of that in Hudi already provides a very rich data management functionality.

I think a lot of companies have the first bucket is companies who just want to build a data team, build this kind of like a vendor neutral, open managed data plane as the sole source of truth for all the data. That, I think, a lot of there are like a large company or option really comes from, I would say. And the second point is a sub point of this, if you look at a lot of these companies, they are also large ML, AI data science companies. For them having some streamlined architecture, where all your operational data like even streams, databases, external data lands in a very clear, schematized kind of format that they can trust on now to kick off all the

downstream analytics and data science. I think Hudi unlocks a lot of these technical capabilities for them.

So, that is what we've seen on the large companies. If you look at our partnership, I think we've had like, at least put together, I think we manage a few exabytes right now, in Hudi with the large companies that have like actually shared the usage. But we also see a lot of small companies picking up Hudi, and that's actually great to see. Because purely even as a project, it offers more tools for you to quickly build a table. So, data engineers allow that, and I want to also highlight that aspect of it because sometimes larger tech companies have a lot of license, right? But we also see organically, smaller companies picking up really trying to build like a hundred table data lake without writing a lot of code. And that is the two segments that we've seen a lot of adoption for Hudi.

[00:21:28] KP: When a company is growing and needs to either start or expand their data team, and they're going to work with Hudi. Could you describe the professional background for the type of engineer who's best suited to manage the system?

[00:21:40] VC: Yeah, I think you need to know, their engineers need to know like good familiarity with the overall broad ecosystem, right? The Hive meta stores, one of Spark or Flink or trade Presto. So, generally familiar with the ecosystem. Then you should be able to pick up Hudi, again, I break it into two parts. One is like if you want to start ingesting data, then they also need to pick up more advanced concepts around like they need to understand how let's say Kafka works. Or if you want to ingest data landing in S3, you need to understand kind of like how like S3 event streams work. And DBZM works if you want to do database since capture. But if you're simply writing EDL, or like data pipelines, then I think with some very foundational knowledge around these data processing frameworks, you should be able to pick up Hudi and get into production very quickly.

[00:22:38] KP: Hudi offers me great opportunities to do both streaming and batch analytics. Could you describe the developer ergonomics of that? What's it like to switch between the two approaches?

[00:22:48] VC: Yeah, that's a great question. So, the one aspect that we haven't touched upon here is the conversion, how Hudi provides you kind of a unified storage layer to mix different ways of data processing. So, when we say streaming analytics, and there is like different shades of real time, correct. We focus on what we call near real time, that it's few minutes. It's still a great improvement from batch jobs or take like ours, but it's not your real time streaming latency where you do less than a minute or even seconds.

So, that's the replay in the few minutes, less than an hour, that's the kind of latency that we set expectations for who Hudi's pipelines. So, if you now take a concrete example, let's say we have, like a very simple task, let me just take an Uber example. There's an upstream database, which is storing all trips. And then the trips have currency in the local currency, like Ubers across many countries, so you just have trips, and then all of their fare values have local currency. And then all we want to do is let's say we replicate it to the lake, we have a table on the lake, which is one to one mapping the upstream database.

Now, if somebody wanted to write a pipeline, where all they want to do is take a small currency conversion table, and then make it all into USD. If that's the pipeline they want to write, with Hudi, what they're able to do is use queries, incremental querying capability, which exposes essentially, you can ask the Hudi table, give me everything that's changed after this point in time. It will hand you all the new upgraded trips, and then you can simply join it with the other table, which is more static, and observe a downstream table, which now keeps the fair values in USD. Let's say like two days later, you realize that, oh, these fare conversions are now wrong, and I need to rebuild a stable, you just switch into batch mode. What I described just now, right before, is what we call incremental mode, which kind of brings stream processing sort of like programming model. But right on top of lakes with no like Kafka or anything involved.

Going back, if you now, for some reason think that, oh, this data is wrong, I need to rebuild the table or backfill it, using the same Hudi table, you're now able to write a batch job, which is just like a SQL or anything else, to again, overwrite the entire table or only selected partitions. So, Hudi gives you the flexibility to makes the more efficient incremental computation, which is mostly right. But also, switch back to the batch mode, without any data moving outside to external systems or anywhere whatsoever, and be able to do the backfills and rebuilding of the table.

[00:25:50] KP: Very neat. Well, for someone who's considering adopting Hudi, can you share a few details on the resources and community sources that are available to help them get onboarded and ask questions and things like that?

[00:26:02] VC: Yeah, sounds good. So, we have a really great community, first of all, in Hudi. I want to just use this also an opportunity to give them a shout out. This has been the best open source experience for me having worked in open source for 12 years. And yeah, you can always start with Hudi or apache.org, our official site documentation, explains and gives you like a good overview of what Hudi is about. How do you use all of the different things that are talked about even in this call. Then if you have questions, there is a Slack channel. You can join the Slack channel, just like self-serve, sign up. And there'll be someone from the Hudi project management committee or committers, contributors, other community members, they're out to answer your questions. We also hold weekly office hours for us to answer any like more one on one questions or things that need more interactive response. And there is the Hudi dev's channel, if you're a developer. You start your developer discussions there. There is also an users list, also, most people prefer Slack for users to post questions.

And if you're using Hudi, and you want support, you can go to the GitHub issues, and you can just file a support issue. Again, one of our members in our community will look at it. Sometimes this could even be issues with your cloud provider. If we can, we'll try to like you know, route to them. Or if this is like Hudi open source issues, we try to like create issues out of them, reproduce them, and then try to get it towards some expertise. So, these are broadly what we do. And of course, we have a monthly community event where we give the community an update on the roadmap, the progress towards the next major release, and also, users can actually come and present cool things that they're doing at Hudi, so that we can all ask questions and learn from each other.

[00:27:58] KP: Could you tease any of the items that are on the roadmap that users might see coming out in the near future?

[00:28:03] VC: Yeah. We're actually working on a pretty large major release in March. So, we plan to add a complete sort of like multi model indexing capabilities. We are streamlining a lot of

our existing indexes, adding new index types to improve performance and query performance. And we are also working on new connectors and improvements in Presto, Trino. So, there's a lot of really cool things, a lot of hardening around space curve, data optimization implementations. So, there's a large release coming out, which we believe will be like really great for users who are looking for updating or update heavy workloads on top of Hudi.

[00:28:53] KP: And how does that approach improve the update workflows?

[00:28:57] VC: Yeah. So, with more indexing, for example, we'll be able to – let's go back to the same example. Okay, there are new chips coming in. And then your data lake table now has chips from all over history from 2010. So, you need to quickly be able to figure out where a given record should go into. So, without an indexing component, you have to scan the entire table, which as you can imagine, will only get slower as we go, and it's practically impossible for any large tables, right? This indexing schemes will be able to quickly figure out how to merge this record and how to reconsider and absorb this update into the table. We are also building – if you take like workloads like user table, there is no pattern whatsoever, right? There are no like temporal patterns. The data is really from a database perspective. It's purely random, right?

So, currently, for random right workloads, we have an external index implementation using Couchbase. While that works great, a lot of people cannot afford to run Couchbase by themselves. So, we are also trying to add a built-in implementation, which keeps the data right on Cloud Storage, and even no external dependencies. So, even for those kinds of random write use cases, I think these features are going to fundamentally improve the right performance.

On the query performance, we are designing like an indexed kind of column stats implementation where we've seen this with lots of large users, if you take by dance or TikToc, they have 10,000 columns in a table. So, none of the existing approaches that we see scale really well for tracking, let's say, a petabyte scale table that has 10,000 columns. So, we are also landing like more advanced/efficient column stats, indexing schemes, where if you now write a query, the query needs to figure out what files I want to scan, like going back to our BI workload, right? So, that is going to fundamentally speed up query planning for small and large column tables. So, we are really excited for all these different kind of secondary indexing

schemes that we're adding to Hudi. I think for me, we always thought about it as a database problem, and then like having these things also brings us closer and closer to what generally how databases are built. So, it's a pretty good moment, I think, for Hudi as well.

[00:31:33] KP: Well, I'd like to pivot slightly and ask you a few questions about Onehouse. Hudi is an open source project. There's the community you described, and documentation online. I could stand it up on my own, why do organizations reach out for help?

[00:31:46] VC: Yeah, so going back to your earlier question, you still need to hire those data engineers that I was describing with that profile, and they're not easy to find. So, what we've seen routinely in the community is that it still takes time for folks to – engineers have to learn, pick it up, operationalize, and also be on call for it. You need to be able to do maintenance on the pipelines. It still takes a lot of time for people to operationalize their legs. So, we feel by providing a more managed approach, similar to what you see with like a Fivetran or a Stitched feeding Cloud Data Warehouse, I think we'll be able to fast track the adoption of the lake and then improve the time to value.

That's the core value prop for Onehouse, and we believe also there are parts of Hudi where even though the code is out there, that there may be workload or table specific optimizations that need to be applied. And some of these standardized data infrastructures, usually typically hard to build in an open source setting, where different companies use different systems for monitoring, logging, and all of that. So, there are also some features that we believe we can build, if we build it as a managed service in a more opinionated way, rather than supporting a lot more flexibility, or continuously for open source DIY adaption.

[00:33:27] KP: When you think about the companies who've adopted it so far, and maybe the companies that haven't. Data lake is a relatively new idea, but certainly a popular one that people have been adopting. Do you have any sense of where an average company is on the maturity lifecycle? Where do they stay in terms of being up to date and best practices and things along those lines?

[00:33:46] VC: Yeah. This is a large part of that I spent my time before we decided to do Onehouse, so I can share it, from my learnings from that perspective. So, what we typically, I

think, I know, completing this in broad strokes, but like a highlight – if we are abstracted and distill it into a single story, I think most folks today, when they're starting out, they start with a fully managed cloud data warehousing stack. Because at that point, their use cases are simpler, just like BI analytics, and then they are also resource constrained. They don't have the time to hire data engineers to build anything out. They just want more pre baked solutions. And after a point in time, I think they either hit cost issues, if the data grows, like the business really picks up, and they feel like, “Oh, we need to invest for the future. We need the more open standardized vendor neutral lock and all these aspects come in.” Or they have genuine needs to, let's say, build a data science facing features like for example, predicting ETS or like analyzing trades in Robin Hood or things like that, right?

So, one of these three reasons get them to build a lake. But from what we see there, since it's so hard to build, even when they say they want to build a lake, it's in a lot of cases, it's still piecemeal. Because some of the data sits in the barrel still. There are even blogs that are now coming in, like most data scientists will tell you, “Hey, okay, you want to run a growth campaign to improve the adoption of your app? Let's bring in even streams that tell us what users are doing. What are they clicking on? What are they seeing? What are they scrolling?”

This data usually is pretty large to send it to a warehouse. So invariably, they end up with this kind of like data scattered across like a warehouse, and some like cloud storage buckets. And data engineers pick and choose and build some tables. And that's where a lot of the pain that we see, if you read about data lakes and data swamps, and all of this general, kind of like fodder around data lakes, I think, is routed around, kind of piecemealing, this architecture this way. This is where we believe, while something actually provides the technology capabilities, now you can replicate your databases, not just to the warehouse, but you can also replicate it to the data lake very quickly. So, which means you can now absorb all kinds of data. You can have like standard quality tables, but organizations still need to make that investment to make it happen. And then should be incentivized to do that yearly on.

So, that's where I see the landscape to be. I see lot of larger companies, lots of like large tech companies are falling into that category, right? But for them, the lake is the main central piece. There are parts of the data where they need premium performance, where they would probably

use a warehouse, but lake is a central piece for most of these people. But if you approach it from the lower and like the smaller – going from small to big, it looks very different, I would say.

[00:37:05] KP: Makes sense. Well, Vinoth, thank you so much for coming on Software Engineering Daily and telling us about Hudi and sharing your experiences.

[00:37:13] VC: All right, it was a pleasure. Great questions and that's why I'm super happy to be here again. So, thanks for having me and I'll see you around.

[END]