A.11 Expert K

Interviewer
I'm *, 24 years old. I have a background in industrial engineering. That was my bachelor. And now I'm doing a master in data science and entrepreneurship. And I'm doing this master, well, I'm doing an internship now. And at the same time, a master thesis at Bain & Company. And the first part of my internship was in Amsterdam. And now I'm located in Chicago to work more closely with the data engineers over here. So I'm really fortunate to be here and really excited to deep dive more into data mesh. The goal of my thesis is to create some frameworks that provide you with some guidance in your data mesh implementation. So my first framework is about data products. The second one will be about the self-serve platform and the third one about the governance principle. And the reason that I included the domain ownership part and the data products part in one framework is that my supervisor and me as well, observe the data products as something that is interchangeably collected with the domain ownership part. So they have to be together and they cannot coexist. Yeah. It's not two separate things. So you can't have data products without domain ownership. So that's why we included this into one framework and that's where we will deep dive more into today. So yeah, that's a short introduction as well.

Expert K
Sure. So let me let front tune. I am having a background in business informatics and transport and supply chain management and information and knowledge management. So I did two masters. Then I worked for respective companies first in the third party logistics environment, then in the medical technologies environment before I really deep dived into the data environment. Let's say it like that the way. That said, I'm now already working for a couple of years for a big software vendor for a cloud data platform. And that's also why it might be that I'm with a couple of things and you will also notice that, I mean, the more people you are talking to, you will always have, well, it's like bias, right? Because people have certain preferences on how to do things. You have engineers that are more into like, okay, we really prefer doing everything like by ourselves. And then you have others that are more into like, okay, let's keep it as simple as possible. And that is actually something that is really coming back in the whole data mesh topic. So talking about that one, I'm already working with companies now since nearly the beginning of, well, data mesh really becoming big, which is around 2019. Since then I'm working with companies on that topic and it developed a lot to a certain degree. To other degrees, there are still a lot of question marks open that need to be answered in the future. But yeah, well, I mean, we will see it's still something that is relatively new and there are not that many successful, like really successful implementations yet. That's also something that you should always keep in mind is like you have some very good examples, but they all face certain difficulties. Like and that makes it again, super interesting, but I guess that's always, you have it everywhere. If you talk about centralization, decentralization, because that's, I mean, you know what, it's what it is, right? Or you're going away from a centralization towards a decentralization and that has both advantages and disadvantages. And most of the times where, what I personally see and it's something that becomes also, I think interesting for you in the later stage, if you're continuously working with that, most companies will go more into a hybrid approach. So they will do like for certain parts, follow the data mesh principles, or it's really keep a centralized approach.

Interviewer
Yeah, that's quite interesting because during the interviews, I observed some conflict in opinions as well. Some people are really into the master data management approach and want to

keep things as central as possible. And others were like, yeah, we should align with the data mesh principles and keep everything so federated as possible, as federated as possible to make sure that the data products can be autonomous, et cetera. So, yeah, but you just told me it's really, yeah, I resonate with that a lot.

Expert K
That's good.

Interviewer
So yeah, let's deep dive into the frameworks because the first one we see over here, this is really like the high level overview framework. So I observed six decisions you can make during your data product implementation. The first one is about what type of data products can be developed. The second one and the third one can be made at the same stage, let's say. So which approach is chosen for the creation of a data product? So do you want to do greenfield development or do you prefer to migrate from your legacy architecture? At the same time, we can think about the outside layer of the data product. So how does our data product communicate with other data products, with the self-serve platform, government layer, management layer and consumers. After that, we can deep dive more into the data product anatomy. So what's going on inside the data products. And we have a decision on the data product interface / contracts. So what kind of ports do we need on the perimeter of our data products, let's say. And the third one, of course, is how to deploy a data product. So if we start with the first one, I observed three different, let's grab a point or a laser point. So this is the decision over here, what type of data product can be developed. These are the three options and everything around it is more like an extension of these three options. So we can expose the data product as raw data. Some practitioners prefer this, but others are really against it. So I think you always should keep some kind of source of truth. So I think this is really important to be able to, yeah, it's more like a source aligned data product, you know, so exposing your data product as raw data. Another option is to expose your data product as derived data. And the third option is to expose your data product as an algorithm, which can be an AI / machine learning model or more like a BI tool and optimization based decision support system. And the hybrid product over here is more like we have a data product and we want to expose raw data as well as derived data. So we have different output ports so we can expose both of them. Data product is more like some kind of, it's a merge of derived data and an algorithm, for example. So both of these aspects are included over here, are merged over here.

Expert K
Okay. So actually expose data product as a composite product should be further on the right side where you have really like the consumer oriented or more like consumer products. Because, I mean, you already say it, right? Expose data product as a composite product. So you already have products before you are actually creating that product.

Interviewer
Yeah. These frameworks are created in Python because I expect some iterations before they can be considered perfect. And if I make all these frameworks in draw.io, I need to construct them over and over again. So that's why I did that. And yeah, it should be more on the right. That would be more convenient, but it's more about the idea.

Expert K
Yeah. I find it very interesting. So you say that you had conversations around, okay, exposing data products as raw data. And you heard so far that this is not really the common practice.

Did they give you like more input on why not?

Interviewer
Yeah, I guess they meant more that you shouldn't expose raw data to the consumers. So that the consumer should not be able to see data as raw because yeah, that will be really messy, et cetera. And it's not convenient for your customer to observe raw data. So I think they were not really conscious about the source-aligned data product because in terms of source line data product, there is a step up front before. Yeah. Before it's...

Expert K
That's why I'm asking, right? I mean, if you're talking about data products, if we really have the... So what is a data product, right? I mean, a data product is a mixture of data plus metadata plus code and then the infrastructural dependency. So you also already have them in there. So if you get data, well, it can be just raw files. It can be views and tables or that kind of stuff. So in that respect, yeah, I totally agree with your separation there. It also needs to have metadata. So it needs to have... Include the technical metadata of its data objects, such as table names, column names, data types, or file format definitions that can also be. So that's why people that are saying... Or if someone is saying like, okay, well, then they definitely leave the whole source oriented data products a little bit out of it. And that's the second part that we always have to think about when we talk about data products is, okay, what... It's not only that you have to have data metadata code infrastructural dependencies. The other part is like, okay, what does a data product need to adhere to? So it needs to be secure. It needs to be discoverable. It needs to be addressable, blah, blah, blah. Yeah, the product seems to be... And one of it is valuable on its own. So as soon as you have raw data that can be valuable on its own, and that's happening quite often. I mean, you can use it for your machine learning algorithms. A lot of data scientists, they love just having directly access to the files instead of going into the structured tables or whatever. So therefore, yeah, it's one of the things where I'm just like, well, it's a very nice one. Raw data. And it's definitely, yeah, on the source oriented data product side.

Interviewer
Do you think I'm missing something here that you can expose your data product in a different way or algorithm derived data and raw data?

Expert K
No, I think actually it's a good way of bucketing it. Because you have also the product can also just be code, right? And you have that. You have that with your algorithm part there. And that's why at the end, I have not seen in practice any other ones. Let's put it that way. So I've always seen like you have either raw data, you have as data as product, or you have code that is really defining it, algorithms that you can share, or you have more structured, for example, tables, views, whatever you want to have. And that's what I understand, but that's what I'm saying, right? That's what I understand under your derived data. Yeah. It could be that you say like, okay, that's not it, but that's what I would understand under it. And then you would have on the consumer oriented side, you really would have then again the composite products that are using whatever you have there before. So if it's like raw data and then maybe together with an algorithm, you create something new that is being used. I mean, that's what you have quite often if you look into marketing analytics, for example. I mean, they are sitting more on the consumer oriented side and they want to use then whatever, like some clickstream data together with a nice algorithm to do brand marketing. Oh, nice. I think it's a good way of putting it.

Interviewer
Great. Yeah. Because this is the first decision. After that, I was thinking more internally. Let me get to the other slide.

Expert K
So but you mean like this decision really? Okay. If I now want to start with the data mesh approach and I look at, okay, how do I define my data product? This is done really, okay.

Interviewer
Yeah, it's more a conceptual choice, but at the same time it has a lot of influence on the other choices as well. Because if you want to expose your data product as derived data, the other decisions need to be different. You need to focus on different parts to expose your data as derived data instead of as an algorithm, for example. So I think it is more sequential. Everything has influence on each other. Yeah. Every decision.

Expert K
That's why I was a little bit struggling with the term of like, okay, decision tree in the first part. Because you can have more and more at the same time, right? Yeah. So that's why it was a little bit...

Interviewer
It's not mutually exclusive in these. Yeah. Do you see my second slide, by the way? I'm not sure.

Expert K
Yes, I do. I do. It's like this approach is chosen for the creation of a data program. Okay. For the creation.

Interviewer
Okay. Because this is more about migration, but if we look at the four migration options, you can choose to do all of them during your implementation, right? You can do master data management as well as zero trust, which is both really important, I guess. And do you know what Strangler Fig is? Strangler Fig is about slowly decomposing your monolithic architecture. So we just pick one of these surfaces out of this monolithic architecture and create a data product out of it. And we do this in certain time periods. So we slowly decompose our monolithic architecture and we slowly build up our data mesh architecture. That's really about Strangler Fig and CQRS is about segregating your read and your write function to make sure that if something is read from your data products, there is nothing overridden immediately. And yeah, the Greenfield development, I think that's pretty clear. I observed these four patterns actually to be really important in your migration part. Do you agree with them? Do you agree that we should have some master data management aspects in our data mesh, for example?

Expert K
Oh yeah, definitely. No question to that. Master data management is always important. And on both levels. I mean, you have it on the local level, right? But that is also one of the things where companies really need to sit down together and really make a decision of like, okay, how can we best structure it that we have like a local and a global level? Because again, there, you don't want to have one part of the company saying, okay, a page is out of a book and the other one is saying like a page is a website. So in that respect, yeah, that one definitely.

Interviewer
Yeah, and Strangler Fig is more about, yeah, slowly decomposing.

Expert K
But I find it interesting how you put them together because, so explain me shortly because you say, okay, which approach is chosen for the creation of a data product? Yeah, I should rephrase that one because now it sounds like there needs to be one choice over here. Exactly. Yeah. So, and this is also, I mean, you always need master data management. It doesn't matter if you're going to pick certain products out and say like, okay, we kind of build a domain around that one, see if the domain only has this one product or it's going to have multiple products that needs to take care of. But the master data management, it will stay because this is really something that you, doesn't matter if you only do a global or do a mix or like do really follow with global, but you always have master data management in it. So I would say actually that this, what you explained me now, how you are defining the Strangler Fig, which is for me, like just, yeah, slowly decomposing the monolith. So this is for me really a migration approach. Like, okay, how do we migrate into a data mesh architecture? Same for Greenfield. I mean, like you're saying like, okay, we just see what's really needed. But that means Greenfield means you first pick a domain. They're like, okay, here, please domain, here's your self-service platform and start creating your products that you need. And then see if you also are responsible for data and data products that will be needed in a further downstream company. Yeah.

Interviewer
Yeah, I agree. So you mean that the master data management, CRM trust architecture and CQRS can be more like child nodes of the other two. So that's more like deep dive patterns. If you already chose the Strangler Fig or the Greenfield development parts.

Expert K
Yeah. Yeah. Yeah. Because they are more, yeah, CQRS, as you say, like read, write, that is really more about like, okay, how do you access then the data products? Yeah. Same zero trust. And so I would actually say like, these two are more about X axis of data products. And master data management is really more the overarching understanding of it. Yeah. So how do we make sure that we are talking about the same stuff? Yeah. So it also falls then later into the interoperability because I mean, or like, it's actually a mixture of not interoperability that much. It's more like discoverability to make sure that it's really understandable.

Interviewer
Yeah. Do you think there can be more patterns during your implementation or are these the most important ones?

Expert K
As I said, like, if it's really about like, okay, how do we go about creating? I think this is really, I also like, at least that's what I see most of the times in the field. Yeah. So it's or companies really decide actively on, we create everything new, or we do like a slow approach. You could also, but that's a little bit dependent on the company and the structure, right? Okay. You could also always say that because companies already have a certain structure, which is most of the times defined as business units, business departments, whatever. Yeah. You could also say, okay, we just theoretically think about like, okay, all these different units or different departments are now own domains. Just say like, okay, now you're responsible for your own data products. I mean, you would probably not do it. Let's put it that way. But you could

theoretically think about it. Yeah. And see further on like, okay, does it make sense to have them being responsible for it? Or do we need to kind of move data out, data in, merge some of them, split them further apart? So I wouldn't really say like, okay, is really something different than your strength La Fig? Where's that term coming from?

Interviewer
Yeah, really interesting. I don't know because strength La Fig, if you think about strength La Fig is just like a jungle, you know, a jungle of leaves, et cetera, strangled in each other. So I don't know where the term comes from, but it's express the migration approach really well, I guess. Yeah. But you just pick one service out of your monolithic architecture, each, each certain period of time.

Expert K
Yeah, it's interesting because I just, again, like, well, decomposing your monolith is definitely something that we talk about. Yeah. And really as the term Strangler-Fig, it's like, yeah, it's like this typical thing of academic writers. Yeah. So throwing a lot of nice terms into it. Nice. Yeah. Yeah. Make sure you are, you're explaining it.

Interviewer
Yeah. Yeah. Because I showed you only the frameworks over here, but I will definitely write context around it. Yeah. Make it more.

Expert K
Oh, good. No. So, but then again, like I said, I think like these two approaches, either slowly decomposing, really seeing what makes sense and going from that route or just building everything from the ground up. This also most of the times then goes hand in hand with your, not only your organizational structure for sure, but also, and that's where it comes really back into it, like the whole self-service platform. And that's why, yeah, it's, you could take them as the overarching ones, I would say. Maybe you could think about taking a third one and just say like, okay, you just pretend that you are in a data mesh situation and then go from there. But I wouldn't, I wouldn't really advise you to put that in.

Interviewer
Okay. Yeah. We can perhaps quickly go on to the next one in terms of time. Can you see the other slides? Okay, great. Because this one is more extensive. This is actually what's all going on outside of your data product. So we of course need a schema registry to make sure that if there is a change event in our data mesh, this change event is converted into a more user-friendly format. So that's what the schema registry takes care of. We need a central data product catalog. Yeah. To centrally manage everything, to govern all the data products, et cetera. And the event streaming backbone. So this can be Kafka, for example, where we make sure that, yeah, data products can subscribe to a Kafka topic. And if something is changing inside the data mesh, every data product will be notified. It can be a shared storage. Some practitioners mentioned that we should have an internal storage, but others mentioned that we should, an internal storage to make your data product as autonomous as possible. But others mentioned that there should be a shared storage because if we have like 10,000 data products and we, yeah, we have internal storage for each of them and we want to check, we want to check them with a federated query. This federated query has to go along all these 10,000 data products, which would be very costly. So that's why they opted for a shared storage. API invocation. Yeah. This is more about how do we access the data product for the consumer? So it can have the rest API, graphQL, gRPC, and on top there, the storage read API and

cloud storage APIs. Those are more, yeah, variants of the tree on the bottom. So this can be a rest API. This can be a graphQL API, et cetera. Of course we need a SQL access point because most data analysts are very familiar with SQL and on the bottom we see more like non-functional requirements such as security controls, a query catalog, a query catalog is, it's some kind of manual where the data analyst can observe all the possible queries he or she can use to access the data. So it's more like a query. Yeah. It's more like a manual and we can have the, yeah, some kind of in-memory cache. So these options I observed during my gray literature research and gray literature research is more about all the sources that haven't been officially published in academic journals. So this can be YouTube videos, medium posts, white papers, things like that. So this is what I observed during my research. I'm curious if you recognize some of these patterns or practices in the field.

Expert K
Yeah. So when I look at that, right, I mean, it is this, this is totally dependent on how you're building up your self-serve platform. Yeah. Okay. Because your, your data product I mean, it also depends on how you are creating data products, but that again is dependent on what is your self-service platform. So if you're looking at the bigger cloud data platforms that can handle already like everything from, yeah, storage, distributed access to data, and also exposing data products. And they are not talking, I mean, you have things like Databricks with more also they're sharing capabilities. Then you have platforms like Snowflake that are very extensively used for that one, right? Even like the AWSs are using now, okay, it's been more of a combination of different tools again, but also they have been data lake, data warehouse, data sharing capabilities. So I'm really dependent on these because they have so much of an influence on how you create a data product. I yeah, I think this one is going to be a hard one for you in the sense of like giving here a general overview of how things need to be done. So because yeah, schema registry or like yeah, change data capture, lineage, access history, all these kinds of things need to be available for data products. No question to that. How do you do that? Well, dependent on the self-service platform.

Interviewer
For example, the event streaming backbone is not a mandatory part of your data mesh. It can be an option if you really like to have everything in real time, but we can of course also focus on batch processing and then we don't need something like Kafka for example.

Expert K
Yeah. And then still, but then still you want to have a programmatic way of making sure that whatever is coming afterwards on top of the data product is always up to date. And again, then yeah, well you could say like, okay, well maybe you need to have a notification service for that one. But if you have, for example, a platform, so now the bias is coming in, right? I mean because if you are using a platform like Snowflake, then you can say, okay, I register here my product. I have a stream on top of that one that is automatically writing metadata about every kind of change so that if there's change happening, you can then use also as a further product that is built on top of that one. You will also have their direct ability, the possibility to put a stream on and see like, okay, what's happening? Why is it changing? Where's the change coming? So then you don't even need like something like a Kafka. So that's why I'm a little bit like, yeah, well it really depends on the self-service platform that you're choosing.

Interviewer
Yeah, I agree.

Expert K
That is also the other part with it is like, that's why I'm saying like there are not that many successful stories out there yet about data mesh because a lot of companies are trying to build it with really own tech stacks for their different domains. And as soon as you do that, you are ending up that you need so many different products to actually be able to provide things like, for example, like, hey, this data product is updated. So please make sure that that is updated. So even more points of possible failures that you're building in there. So that's why I'm like, yeah, so the event streaming, for example, yes, you could do that, have that besides if you do batches. But what comes with that is really the contracts that you have to build. So like, okay, what kind of contracts do you have for your data product? And that is a big part of it so that you really need to define upfront for a data product. What are the SLAs? Because yeah, maybe you just need batch, maybe you just need streaming. Also regarding quality assurance. I mean, if you're looking into the real product world, you have departments, I mean, you know that, right? I mean, you did like engineering. So you have departments that are only busy with checking like, okay, are the products that we are producing here and if it's just a nail, according to the standards and the quality that we have to have. And there are certain standards that I put on it and that's all stuff that needs to be done here. And then if you have that, well, API, yeah, that's for me, like all of that is really like platform dependent. It's not that much product dependent.

Interviewer
Okay, so you should include this one in the self-serve platform frameworks.

Expert K
Okay. I would say so because here you're really laying down like, okay, what is the requirement of a platform to make sure that the, like part of it, right? I mean, because again, for a product to really work, you need to have lineage. You need to have stuff like access history. You have it down here. Also with X control, all that kind of stuff. So it's all stuff that falls under the product itself, but then the interaction between the product and other product that is really dependent on the self-service platform. Because also there, I mean like a product that just imagine you are mainly like a domain that is working with Python, right? So you're doing everything in Python. You're like creating all your products in Python, and then you want to kind of make it available to another team and that team is only working with SQL. And then you also already need to start thinking like, okay, so how do we actually build the product that either it's compatible, that they understand it, that they can further use it, or you have a self-service platform that says like, I don't care if you do it in Python or SQL or whatever, I translate everything into language afterwards automatically so that I keep the, sorry, like the interoperability. That's what I wanted to say.

Interviewer
To make your output board more like multi-model, right? That's how Zhamak Dehghani described it in her book. Yeah. Really interesting. Yeah. I will definitely have a thought about this one to perhaps include this in the self-service platform frameworks. So we can perhaps now deep dive more into the data product itself. So this is really what I observed to be in the data product. So insight data product parameter. Yeah. If we talk about events streaming, of course we need something like a change data capture. This can be really important to be connected to your event streaming background. Immutable change audit log. This immutable change audit log is actually some kind of database where you store all your change events. So if something is changing, the change data capture notices this and creates change events. This change event goes through the event streaming backbone and other data products are

subscribed to this event streaming backbone. And this change events will be stored in each immutable change audit log of each data product. So they all have their immutable change audit log and it's immutable because it doesn't change over time. And it's more like an append only mechanism. So every change event is appended in this data storage. So that's what I mean with immutable change audit log so that all your other data products know about these changes or know about the changes that happen to the data. Yeah. So we can of course have internal storages, which is the opposite of the previous one in the outside layer where we can have shared storages, a data catalog to make your data products autonomous. And if a data scientist wants to check what's inside the data product, he or she can just have a look at the data catalog where all the metadata stores. We can have an observation plane, which is for the data scientists to observe the data quality inside the data products. We can have a control plane, which is meant for the government team to make sure that all the policies are enforced, all the global policies are enforced in the data products. And the data onboarding part is more about all the transformations that are being applied in the data products to the data set. So it's like Spark and all these kind of things. It's more about the transformations.

Expert K
To be honest, I think you're very having that a lot.

Interviewer
So are all options. It's not mandatory to have a change in the capture.

Expert K
Oh no, I think that's, I'm just looking at the points. I think I actually have, I haven't really come up with anything that you're missing because yeah.

Interviewer
Because this is really data products related, I guess.

Expert K
Yeah, exactly. I think it's very, very, very straightforward. And then also like with the versioning practice, what's happening there. I mean also that one, I mean, a lot of that you have nowadays also with the table formats that are being used. I mean, if you look at Apache Iceberg, for example, then you have the table format there, which is based on exactly that, that you don't like that you keep all the different underlying files as they are. And you are just adding everything that's new. Yeah. Yeah. So that one actually is a little bit coming back to what you said before with the different practice, right? So that you say, hey, you have some that say like, okay, you should have shared storage, some that say you should not have a shared storage. Then the question is really like, okay, how, and I think that is something where, again, it depends on are you working with legacy systems and you are trying to build a data mesh on a legacy system because then you will end up having like multiple copies of the data to provide it to the different teams as products. And then you really need to kind of set up all these flows of, hey, the data is updated. Please change your source. Or are you working with modern technologies and modern tech stack where these kinds of things are that you can just really have at one time stored and you're building on top of that one. So because I mean, I don't know like how much you are into that stuff, but what you can do is you can really say like, okay, I store it one time. I build views. If you are really talking about a data product, as a view, for example, that is then for a certain group, they can access it. They can query it. They can build on top of that via the access layer where I have made it available. But then it's also going back to your, okay, how do we give them access? Because is it a pull

or a push type of access? So do they really need to have an environment request access or do we say, okay, we have a common access layer where we just give everyone access. So there are also like different ways of how to deal with it. Yeah. Or again, very dependent on the needs of a company because if you have like products where you say, okay, they're not very sensitive, but they're necessary. And why would you make the hustle of creating a access flow where you really are only giving a certain group access, right? Yeah. And then say like, okay, well, we give it openly accessible via an access layer. Yeah.

Interviewer
I agree with that. Yeah.

Expert K
So they're like, yeah, it's a lot of like dependencies on so many. No, but I think actually like when I look at that, I wouldn't now directly come up with anything that I would say is missing. So because you cover with the points, you cover also the idea of lineage, you cover with it again, the access control, you cover with it. Yeah. So. Yeah.

Interviewer
Because the next framework, the, that one is really related to this one because yeah, if we think about data catalog, observation plane, control plane, we can think about the observation port control part and discovery port as well. So these discovery port is connected to the data catalog. So this is more like a sneak preview of all the insights of the data product. The control port is connected to the control planes, where the control plane is more like an interface and that's for enforcing global policies, et cetera. The observation port is connected to the observation plane, of course. And this one, yeah, this one helps you to observe the data quality inside the data products. So, and of course, practitioners mentioned that I forgot two very important ports, the input port and output port, which are the most obvious ones I think. So yeah, in total, there would be five different kinds of ports.

Expert K
Yeah. However, input output are really, okay, how do you get the data in and out? Right. Whereas the other ones, so the ones that you're discussing here, right? I mean, they are really more around the specifications of the data product, right? I mean, okay, how do you find it? How do you make sure it has the right? So this is really about the, yeah, you say it also in the beginning, like the contracts. Yeah. This is also what I meant with what are the SLAs? Yeah. What are the quality agreements that you have? What I just find interesting is like, because if you think in tools, would you say that these three observation control discovery are three different ones?

Interviewer
Different APIs, you mean? Different ports to access the data or what do you mean?

Expert K
Yeah. Different in the sense of like, do you think that if you have a data product, right, you're having it on some platform, let's say it stands in a database and you want to kind of make sure, okay, you have the observation, the control and the discovery port on top of it. So the funny thing about that one is part of it, you could say like, okay, if I just have a database on top of that, I have a catalog, like a Colibra, then you already have the possibility to see like all of that, right? You can discover it, you can have like a shell in it, you have the observation in it. So that's what I mean. Like if you think about it in tools.

Interviewer
I agree with that because one of the practitioners mentioned, what was more from a master data management perspective, you mentioned the fact that we can merge these three ports into one overarching management layer, like you just mentioned, instead of creating these three different ports, because again, you need to create a federated query that goes along all these data products and which would be very cost inefficient perhaps. So yeah, this is again a choice on granularity. How much power do you want to give the data product owner or how much power do you want to give to the central authority? So yeah, I should include that as well, that we can create some kind of overarching management layer that includes these three ports.

Expert K
Yeah. Okay, great.

Interviewer
And we can perhaps quickly go on to the last one because we briefly talked about this one. Yeah, so I observed that many practitioners mentioned containerization as a deployment strategy. And for example, function as a service, I guess that's also a deployment strategy. So I should rewrite this arrow as a second option because Kubernetes over here should be a child node. So observe this as containerization and the second option is serverless as service function. But you think there are many more options in terms of deployment.

Expert K
Yeah. So because again, sure people are using Kubernetes or they're using Docker. So if you're looking for example at implementations where you really say like, okay, well, let's say my data product is a table. I have set up my engineering pipelines for it. I've set up everything so that my table is at the end standing in my database. And I make that one available via whatever. It's a cloud database, so I just can make it available to other users or an end can list it in my data catalog. Would I always set everything up in a container? No, definitely not. I wouldn't because like keeping that one up and running and like being constantly busy with that, no. I would definitely not do that. I would just, yeah, like do it without. I said, like the reason why in my opinion, like it would make sense is really okay if you put really all of the automation of the data product and the maintenance of it really into Docker container. But yeah, again, like creating and managing containers is not a trivial job. It's just not. And that's why I wouldn't go for that one if I would go for a self-service platform. I would always go for a more like, yeah, like a cloud data platform that can give everyone their own distributed environment where they can really work with the data independently, but still be on one, really on one platform. So yeah, that's why I was a little bit like, ah.

Interviewer
Yeah, I agree. This one needs a lot of reconstructing.

Expert K
Yeah. Let me see. I'm going to send you something actually. It's very nice. And it's also, I mean, you know, that ThoughtWorks where Shamak was working and everything. I mean, this one is really going more. It's a presentation I just saw lately from again, Rosh. And Rosh is one of the, yeah, the leading data mesh implementations that you can find. So first of all, it's anyway, like have, have a look at all the content that's out there. It's like, man, they have so much content there on how they have built up their data mesh, what kind of decisions they made. There is Omar. Omar is the, yeah, I don't exactly know his title. I'm really not into

titles, but he has also written a lot of blog posts also on the different topics and how they made decisions. And I think that is, that is giving you a very good insight into data mesh practice.

Interviewer
Yeah. Sounds good. Would love to read.

Expert K
Yeah. I will, I will send you like a couple of things to that one via email because then you have that and I'm very sorry, but I have to drop out. Yeah, no worries. I haven't.

Interviewer
Oh, well it's 9.00 AM over here, but thank you so much for this interview. I learned so, so many new things. I'm really grateful for that. And perhaps we will see each other when we talk about the self-serve platform frameworks.

Expert K
Feel free to reach out. And also if you need other contacts where you see like, Hey, they are like maybe interesting and you see like they're connected with me on LinkedIn. I can also always introduce you. That's not a problem.

Interviewer
Nice. Thank you so much, *. And have a good day. Hope to see you soon.

Expert K
Have fun in Chicago. Yeah.

Interviewer
Thank you so much. Okay.

Expert K
Bye bye.