**0:00  
Alexey**  
Hi, everyone! Welcome to our event. This is brought to you by DataTalks.Club, a community for people who love data. We have weekly events — well, almost weekly lately, but we're getting back on track. Actually, we have two events this week! If you want to check out our upcoming events, there's a link in the description. I think I need to update it since we now have over 50K subscribers, so if you haven't yet, hit the subscribe button.

Also, we have a live chat where you can ask questions. There’s a pinned link for submitting your questions, and we’ll cover them during the interview. That’s it for the intro! I’ll stop sharing my screen now. We haven’t had a podcast interview in a while — maybe a month — but don’t worry, we’ll have more content coming soon. Today, we have Raphaël with us. If you're ready, we can start.

**1:18  
Raphaël**  
Yes, definitely.

**1:21  
Alexey**  
Great! This week we’re talking about MLOps with a special guest, Raphaël Hoogvliets. Raphaël is a leader in MLOps with a background in data science and machine learning. You might know him from LinkedIn, where he frequently shares MLOps content. I see his posts daily — though I suspect his profile picture is AI-generated, right? Anyway, Raphaël currently leads a team of engineers at Eneco, a major sustainable energy provider.

**2:06**Raphaël  
Hey, Alexey. Thanks so much for having me here. It's great to be here.

**2:10  
Alexey**  
It’s a pleasure to finally make this happen! We’ve been planning this for a while. And a quick shoutout to Johanna Bayer for preparing today’s questions.

Before we dive into MLOps, let’s start with your background. Can you tell us about your career journey so far?

## Career journey and transition into MLOps

**2:34  
Raphaël**  
Sure! I’ve been working in the data field for over ten years now. I started as a data scientist, which was challenging at first because there was so much to learn. But I kept pushing through.

**3:02  
Alexey**  
What did you do before that?

**3:04  
Raphaël**  
Something quite different. Before data science, I spent five years in sustainable agriculture. My role included project management and lobbying for a green organization.

**3:24  
Alexey**  
So, it wasn’t IT-related, right?

**3:26  
Raphaël**  
Not initially, no. But I noticed that large projects lacked proper data management. Farmers had a lot of valuable data on crop quality and sustainability, but it was mostly on paper or outdated systems. That’s when I started thinking there was potential for improvement. Eventually, I moved into agricultural innovation, focusing on technology and data, which ultimately led me to data science.

I transitioned from project manager to consultant, freelancing in agriculture and food tech. But agriculture isn’t always data-driven, so I chose some challenging projects. About ten years ago, I was working with a client on a proof of concept, and they asked if I could put it in production. I didn’t even know what “production” meant back then, so I did some research, learned a bit about IT, and realized I needed to dive deeper into these concepts.

This led me to an IT consultancy — a Microsoft Platinum partner — where I worked as a data scientist in a mostly BI-engineering team. It was challenging but a great learning experience. Later, I joined a more data-focused consultancy and became a lead data scientist. Here, I really started learning about production deployments.

In consultancies, projects are often short-term, but data science projects require more time. So, I joined an internal team where I got my first taste of MLOps. We had valuable models, but they were built in a very "spaghetti code" style on an old R server. I got the chance to upgrade this setup, working with a great mentor, and I found that I really enjoyed MLOps. It was exciting to add value beyond just building models.

That was about three years ago. Since then, I’ve worked in MLOps and even started freelancing. I’m currently contracting with Eneco, where I’ll soon join as a full-time employee to continue building the team and see the ongoing reorganization through. There's so much work to do, and it’s fulfilling to stay on and see things progress.

## Dutch agriculture and its challenges

**8:41  
Alexey**  
It’s a fascinating journey! You mentioned starting in agriculture, and given that you're from the Netherlands, I know the country is advanced in agriculture. Even though it’s a small country, I see Dutch produce across Europe. It’s impressive how much you achieve with limited land — it’s almost like an agricultural marvel!

**9:13  
Raphaël**  
Yes, it’s definitely a point of pride for us! In the Netherlands, we often say we're the second-largest agricultural exporter, but that statistic can be misleading. It includes a lot of re-exports. In reality, we rank around 22nd in actual production, which is still impressive for a small country.

Our agricultural success is due to high-tech innovations, from genetics to production processes. However, our approach also has a high environmental footprint. We import a lot of grains and soy from the Americas to feed livestock, which creates challenges like excess manure and air quality issues. But I believe we can continue being a sustainable player if we adapt the right practices.

## The concept of "technical debt" in MLOps

**10:36  
Alexey**  
Back to MLOps — your LinkedIn profile has an interesting tagline: “Creating the future’s technical debt today.” What does that mean?

**11:01  
Raphaël**  
It’s a bit of a joke but also has a serious side. In our field, technology moves so quickly that by the time you implement something, it might already feel outdated.

There's a gap between when you make design decisions and when you implement them. Sometimes, as new tools or models emerge, it feels like we’re creating "technical debt" just by sticking to what’s current. But in reality, true technical debt is the gap between an organization’s capabilities and its ambitions.

**12:19  
Alexey**But it's unavoidable, right? Sometimes you want to create a prototype, move fast, and you're unsure how many out of 10 prototypes will survive — maybe just one. So, you don't necessarily want to invest a lot of effort into each prototype. But when one works, you know, okay, it proved its value, so now it's time to repay the debt.

**12:47  
Raphaël**Yeah, I agree. These decisions and conversations happen daily, especially between tech leads and product managers, but also among tech leads. At Eco, some of the teams have a startup mentality — cut corners, move fast, and focus on delivering value for the customer. Others are more cautious, saying, "Let's do this the right way before we deploy."

**13:21  
Alexey**We should stop cutting corners, right?

**13:24  
Raphaël**Exactly, yeah. It's an ongoing discussion. And in engineering, the cliché is that everything is a trade-off, and that's true on the product level as well.

## Trade-offs in MLOps: moving fast vs. doing things right

**13:37  
Alexey**You mentioned building teams in your career. You’ve built teams multiple times, and now you’re building another one. Why focus on teams specifically? And what do teams have to do with ML Ops?

## Building teams and the role of coordination in MLOps

**14:05  
Raphaël**Great question. It was a natural progression from my earlier life. I’ve always valued teamwork, from playing sports to working in a team at an art-house cinema and even in online games. I’ve always focused on how coordination and good culture are key. When I moved into data science, I saw the importance of having a well-coordinated team. In data science and ML Ops, a strong team is crucial. If you want to go fast, you go alone, but if you want to go far, you go together. In non-tech organizations, doing data science can be particularly challenging, and having a well-rounded team with diverse roles ensures success.

## Key roles in an MLOps team: evangelists and tech translators

**16:58  
Alexey**So, what makes a good team? What kind of people do you need?

**17:29**RaphaëlIt depends on the context and the organization’s maturity, but there are common roles. One often overlooked role is the evangelist. Someone has to advocate for the team, whether it's internally or on the executive level. This person isn’t just about stakeholder management, but about driving the vision and support. In product teams, you also need a tech translator — someone who can bridge the gap between technical and non-technical stakeholders. This could be a product manager, but often the roles are split, with one person focusing more on the technical side.

**18:37  
Alexey**We’re talking about non-IT companies, where ML Ops is secondary, right?

**18:54  
Raphaël**Exactly. In these companies, it’s essential to have someone at the executive level who understands ML Ops, or an evangelist who can rally support. Additionally, a tech translator helps communicate technical complexities in a way the business side can understand. These roles are vital for success in ML Ops.

**19:56  
Alexey**So, evangelist and tech translator are key roles in the team?

**20:33  
Raphaël**Yes, and it’s possible for one person to fill both roles. On the technical side, having an experienced lead is essential — someone who understands ML Ops principles and can guide the team. It’s great if you can split between ML Ops engineers focused on building infrastructure and automating the ML lifecycle, and ML engineers working with data scientists on the product side. But the team composition really depends on the type of ML you're working with.

## Role of the MLOps team in an organization

**23:01  
Alexey**  
So what does the ML Ops team do? It sounds like a central team that helps other teams, is that right?

**23:32  
Raphaël**Yes, it depends on the organization, but in many cases, the ML Ops team is centralized. In our agile framework, we act as an enabling team. We work closely with ML engineers, helping define best practices, create design documentation, and build reusable tools. We focus on deployment, maintenance, and monitoring, which are key components of ML Ops. We aim to make things easier for the other teams to adopt, but we also have to make sure we’re flexible tnough not to alienate them.

## How MLOps teams assist product teams

**25:19  
Alexey**  
So the ML Ops team helps 34 product teams with different use cases, like demand forecasting or energy supplier maintenance. Is that correct?

**25:20  
Raphaël**Yes, that’s the setup. We have product teams working on different use cases, and the ML Ops team helps by providing infrastructure, tools, and best practices to make model deployment easier. It’s a luxury to have both a centralized ML Ops team and ML engineers embedded in the product teams, which is the situation we're in.

## Standardizing practices in MLOps

**27:56  
Alexey**How do you standardize practices when only about 25–30% of data scientists are on board with your framework? How do you get the rest of the team to follow?

**31:07  
Raphaël**Iteration is key. You test, talk to people, and get feedback. It takes time to build relationships and trust. You need to listen to your end users — data scientists are crucial in this process — and adjust your framework accordingly. You can’t please everyone, but you can find common ground. Some aspects, like CI, repository structure, and packaging solutions, are easy to standardize. But the actual code structure is trickier. You need to find a balance between enforcing standards and allowing flexibility for data scientists to work the way they’re comfortable.

## Getting feedback and creating buy-in from data scientists

**32:46  
Alexey**How do you go about talking and getting feedback? Do you select a few projects that are either the most important or not so important because you don’t want to touch the important projects? Walk us through the process of understanding what kind of standards we can have as a team and what kind of standards will get adoption.

**33:13  
Raphaël**From the data science side, in my current situation, we're in a position of luxury because we have many ML engineers. But if you don't have that, and you're just an isolated AIOps team, I would approach it like a product manager. You treat the process itself as you would when developing products for users. You try to develop your AIOps platform that way for developers. People often talk about user experience, but developer experience is a huge driver for success. While I'm not a product manager, I do apply some of those principles for AIOps.

To create buy-in from the organization, the first thing you do is collect pain points — what are people struggling with? Then you can create a matrix to compare what I, as the ML Ops lead, think we should work on versus what the data scientists think we should focus on. In this matrix, you’ll have four quadrants. Start with the one where there's overlap to create quick wins, even if you don’t think it’s the most important thing to do. The key is to build trust and prove your value.

Once you have the pain points, you also need to show a clear "before" and "after." Before starting, paint a picture of where things are now, acknowledge that it will take time and possibly some of their regular work, but highlight the clear gains once it’s done. These could include saved time in deployments, mitigated risks, or saved money. Once we save those resources, the data scientists will have more time for the actual work they love, rather than getting stuck in operations. Without proper machine learning operations, data scientists often end up debugging pipelines. As they build more models and solutions, they get criticized for broken pipelines. Eventually, they start manually running chunks of code with print statements to figure out what went wrong. This process takes a lot of time. You need to show the team that by implementing these tools and practices, they can return to improving models or building new AI solutions rather than fixing pipelines.

## The importance of addressing pain points in MLOps

**36:55  
Alexey**So that’s why you start with pain points. As a data scientist, I definitely don't like debugging my pipelines. You identify what they don’t like, which avoids the risk of, as you mentioned, a platform engineer thinking the best way to deploy models is the only way, and then others don't find it useful. You could end up with fancy solutions that aren’t solving the real pain points of the product teams.

**37:32  
Raphaël**Exactly. You might start rolling out your amazing practices, and data scientists will find that the tests don’t accept what they just built, or even worse, the pre-commit hooks don’t work. They’ll ask, "What is this mypy thing? I can't even merge my code now." And if they’re using branches, it gets worse. That’s how you lose buy-in. You really need to show the value of what you're doing.

From the leadership side, one of the challenges in ML Ops is dealing with KPIs and OKRs. Everyone wants measurable results, but it can be tough to show those in engineering, especially when it feels like, “If we weren’t here, nothing would work.” But you need to present those KPIs, even if it's a challenge.

**38:41  
Alexey**It could be as simple as tracking the number of models deployed through the platform.

**38:44  
Raphaël**Exactly.

**38:46  
Alexey**That’s a good way to measure. As we discussed earlier, you need to approach an ML Ops team as an internal product team. You need to talk to users rather than assume what they need.

## Best practices and tools for standardizing MLOps processes

**39:06  
Raphaël**For sure.

**39:08  
Alexey**We briefly touched on best practices and tools. You mentioned having proper CI and a clear structure for ML repositories and packaging. Are there any other must-haves when it comes to standardization and best practices?

**39:41  
Raphaël**It's always a good idea to isolate your parameters, especially when building solutions. For software engineers, this might be a no-brainer, but for data scientists, it's crucial.

**39:58  
Alexey**And this should be done in a standardized way, right? Not one team using YAML files, another using JSON, and a third using XML. Everyone should stick to the same standard.

**40:10  
Raphaël**Exactly.

For your testing suite, it’s difficult to get code coverage for everything, but I think testing data transformations — preprocessing and post-processing — should always be a priority. And your development team can handle this themselves.

Another thing to consider is data exploration. A lot of knowledge is captured there, and it can help with monitoring setups. Even if you don’t have an advanced monitoring system yet, the insights gained during data exploration can be valuable. Before, we’d just comment out code for visualization, but if your code is in production, you want to keep that part in. This will aid with root cause analysis down the line.

**41:58  
Alexey**So, it’s more about organizing your code — keeping exploratory work separate from deployment code, and having clear test coverage, right?

**42:13  
Raphaël**Exactly. But there’s a risk that exploratory work gets lost. For example, code could be version-controlled, but the original exploration might be on someone’s desktop who’s left the company.

## Value of data versioning and reproducibility

**42:31  
Alexey**So, if we have a Git repo, should we keep exploratory work in a specific folder, like a "notebooks" folder? Even if it’s messy, just commit it, push it, and keep it around?

**42:54  
Raphaël**Yes, exactly. There’s usually a lot of value there. It’s worth keeping around.

For best practices, once you reach a more advanced stage, reproducibility and traceability become important. Traceability depends on your sector and legal requirements, but reproducibility is key. While it's difficult to make everything 100% reproducible, tying your code to data versioning is a good start. If you know which data version is connected to a particular deployment, it helps you reverse-engineer when needed.

## When to start thinking about data versioning

**44:22  
Alexey**At what point in an organization’s maturity should they start thinking about data versioning? Sometimes it feels like overkill, especially if we only have a few models and aren’t dealing with a large portfolio. Do we need to care about data versioning from the start, or can it come later?

**44:46  
Raphaël**I agree, it can feel overwhelming. It really depends on your sector and any obligations you have to customers or legal frameworks. But, yes, it’s not always necessary to focus on data versioning from the start, especially for smaller teams. It can come later as your needs grow.

## Importance of data science experience for MLOps

**45:10  
Alexey**We have a few questions from the audience. The first one is: is it important to first work as a data scientist before moving into MLOps? And maybe here we can also discuss what MLOps actually means in this context. There’s this idea that MLOps is strictly about tools. As a data scientist, I start using these tools and then move into MLOps. But we've also talked about other aspects of MLOps, like processes and team structure. With this in mind, do you think it’s necessary to work as a data scientist first, or is it not required?

**45:56  
Raphaël**You do need these skills in your team. Not everyone has to come from a data science background, but you do need those skills in the mix.

**46:05  
Alexey**In the MLOps team, right?

## Skill mix needed in MLOps teams

**46:06  
Raphaël**Yes, exactly. It’s good to have a mix of skills and backgrounds. MLOps has a lot of overlap with SRE (Site Reliability Engineering), which some people refer to as DevOps. It’s interesting because I learned not too long ago that DevOps is more of a movement, and no one really knows exactly what it is. On the other hand, SRE is a well-defined practice. MLOps overlaps a lot with SRE, so it’s useful to have someone with that expertise. It’s also helpful to have good software engineering experience. As data scientists, we’re generally not the best software engineers. Interestingly, roles like SRE and platform engineering are often called data engineering in many organizations. Data engineers are often building data pipelines or doing data warehousing, but they also do a lot more. In many cases, adding a data engineer to your MLOps team can be really beneficial.

## Building a diverse MLOps team

**47:33  
Alexey**So, in general, we aim for a diverse set of skills: someone with engineering experience, someone with platform experience, and someone with data science experience. Is it fair to say that while it’s not essential to have worked as a data scientist before moving into MLOps, having that expertise in the team is important?

**48:02  
Raphaël**Yes, ideally more than one data scientist, but you definitely need that skill mix.

**48:08  
Alexey**Probably someone with translation skills should at least have some experience in data science.

**48:15  
Raphaël**Yes, that could definitely help.

## Best practices for implementing MLOps in new teams

**48:18  
Alexey**A question from Sam: what would you say is the best place to start when implementing MLOps in a new team? The team has experimented with Vertex AI.

**48:31  
Raphaël**What is the question again?

**48:33  
Alexey**What is the best way to start implementing MLOps in a new team?

**48:41  
Raphaël**It’s important, from a product management perspective, to think about what’s most needed in the organization. What challenge are you trying to solve? If the problem is that you have models running in production and you don’t know what they’re doing, then you start with monitoring. Or, if the issue is that deploying new model versions takes too long and you want to deploy a new model every month, but it currently takes multiple months, then you start there. It’s a hard question to answer without more context, but there’s always low-hanging fruit in MLOps. For me, CI/CD is always the starting point. You can set that up quickly.

## Starting with CI/CD in MLOps

**49:52  
Alexey**So, essentially, you talk to data scientists and users, understand their pain points, and go from there.

**50:04  
Raphaël**Yes. And if you want to start with MLOps, you should first assess whether you can build it with the tools you already have. If you have the tools, use them. Sometimes people want new tools, but depending on the company type, getting new tools can take a lot of time. In a startup, you can just get the tools with a credit card, but in corporate environments, procurement processes can be very slow. So, if you can, start with what you have. But if you find you have nothing to work with, realize that early. For instance, version control is crucial. If you don’t have it, you should flag it with leadership right away and get a subscription to the right tools.

## Key components for a complete MLOps setup

**51:21  
Alexey**Another question came up as you were speaking. You mentioned building MLOps with the tools you have. What does it actually mean to build MLOps? How do we know when we have a complete MLOps setup? Is there a set of tools we need to have to say, “Okay, we have MLOps now”?

**51:48  
Raphaël**Yes, great question. There are a few frameworks for MLOps. Last year, I worked with Marvelous MLOps on a blog and content platform, and they created a framework called the MLOps Toolbelt. This is the one I like best. It’s based on existing literature and includes a set of components. If you have all of these, you’ve got a good MLOps setup.

**52:33  
Alexey**These components include experiment tracking, model registry, and monitoring?

**52:39  
Raphaël**Yes, exactly. Version control, CI/CD, containerization, model registry, experiment tracking, container registry, monitoring, compute, and serving. Another important component is a package registry. So, you should have a private package registry as well.

## Role of package registries in MLOps

**53:08  
Alexey**Why do we need a package registry?

**53:27  
Raphaël**You can certainly build a Docker image without it, but what I like about packaging is that you can bundle a lot of things together. For example, with Python packaging, you can manage dependencies and configurations. Python packaging lets you configure important settings in the package itself, which helps ensure compatibility with other software. You can define version ranges for your dependencies, which helps maintain compatibility with other packages.

## Using Docker vs. packages in MLOps

**54:12  
Alexey**Why not just use Docker?

**54:16  
Raphaël**You can build a Docker image, but you’re often not working in isolation. Your package may interact with other pieces of software. With a package, you can configure its dependencies, so it will work with other software components. Without that, you risk creating conflicts between versions. You could also set up each component as its own container, but that could add a lot of overhead.

**55:54  
Alexey**I imagine that if everything isn't in its own container — each with its own dependencies like Pandas or NumPy — then when we deploy multiple models, each one will have its own set of dependencies, which can add up quickly. However, if we use packages, maybe all models could share the same version of a dependency like Pandas.

**56:36  
Raphaël**Hopefully, this can be resolved within the version ranges. That’s what we aim for.

**56:41  
Alexey**It sounds like there are use cases where it's easier to use a package rather than build a full container.

**56:50  
Raphaël**We usually do both. We're using Databricks, but we also have a setup with Kubernetes. We build our Docker images using our packages. It really depends on the complexity of your setup. When I first saw it in 2019, someone advocated for separate Docker images for each component of the ML pipeline — one for data handling, one for preprocessing, one for modeling, etc. It seemed amazing because you could easily swap them out when something changes. But the question is, does it make it easier or more complex for teams to be autonomous, and for data scientists to use this setup?

**57:39  
Alexey**I noticed we have some questions in the live chat. Sorry, I was focused on Slido. Do you have time for one more question?

**57:48  
Raphaël**I actually have time. We originally planned to finish early, but I have another half hour.

## Examples of MLOps success and failure stories

**57:56  
Alexey**Great! Let’s go for it. Here's a question from Zanna. She's starting in data management and likes your point about addressing pain points to get people to use your solution. Do you have examples where this approach worked well, and where it failed?

**58:20  
Raphaël**A successful example is something you mentioned earlier — showing the frequency of deployments. We showed the business that they were deploying just a few times a year, and then we showed them that with the new setup, they could deploy every day.

**58:37  
Alexey**So deployment was a pain point for them?

**58:40  
Raphaël**Yes, deployment was taking a long time. Training and compute were also slow, and testing took forever. We convinced them that with MLOps, we could do all of this in just a few hours, and that worked. As for a failure, we built a successful data science solution in a proof of concept for a client, and then sold them a data platform. In 2019, we recommended setting up a data lake, using Databricks to run models. We built and sold all these projects, but where we failed was when they had an integration freeze. The board decided no integrations were allowed. We had a complete data platform, but it couldn’t connect to any data sources. We worked there for two years, but in the end, we couldn't convince them to proceed.

**60:06  
Alexey**That must have been frustrating.

**60:09  
Raphaël**Yes, I had two difficult projects in a row, both just under a year. This was the second one. After that, I took some time off. I also faced some personal challenges, but I essentially burned out on that project.

**60:29  
Alexey**I can imagine. When there’s a decision you can’t influence and they just say, "We're not integrating it," it’s hard to understand why you spent all that time building it.

**60:42  
Raphaël**Exactly. They made us build something that would run on uploaded CSV files, so that’s what we did, but it wasn’t the right approach.

## What MLOps is in simple terms

**60:54  
Alexey**Here’s an interesting question we might have started with earlier: What is MLOps in simple terms?

**61:05  
Raphaël**MLOps is the operational side of building machine learning solutions. You build a machine learning model, and then it starts running in your business. It's similar to other business operations, like HR or logistics — this is the machine learning equivalent.

**61:25  
Alexey**So it’s a set of tools and best practices to ensure machine learning models run smoothly?

**61:31  
Raphaël**Yes, exactly.

**61:32  
Alexey**And to make sure they keep running properly?

**61:36  
Raphaël**Yes, it’s not just about getting them running once — it’s about keeping them running properly. The challenge is that data changes all the time.

**61:46  
Alexey**And by adding "the right way," you’ve made it so much more complex!

**61:50  
Raphaël**Yeah, it’s definitely more complex than other software operations, especially because of the data aspect.

## The complexity of achieving easy deployment, monitoring, and maintenance

**61:58  
Alexey**I think you mentioned earlier that the focus of an MLOps team is on easy deployment, easy monitoring, and easy maintenance. But achieving all three is quite difficult, isn’t it?

**62:15  
Raphaël**Yes, it's a challenging field. You need to know a little bit about a lot of things, and then a lot about a few specific areas.

**62:36  
Alexey**I’ve got your GitHub profile here.

**62:39  
Raphaël**Yes, definitely.

**62:42  
Alexey**OK, I think that’s all for today. You’ve shared a lot of valuable insights. Thank you, Raphaël. I’ve taken tons of notes! I’m really happy we finally managed to record this. It’s been great, and we’ve had a great turnout. Thanks to everyone for joining and for your active participation. And once again, thanks to Raphaël for sharing your experience with us. I hope you won’t face another integration freeze in your career — it must have been devastating. Thanks for sharing that with us as well.

**63:29  
Raphaël**Thank you so much for having me, and thanks to everyone who joined. I’d also like to give a big compliment to you and Johanna for how well you prepare and the questions you put together. I really enjoy this format. It’s been a pleasure to be here today. Big thanks to both of you!

**63:47  
Alexey**The credit goes to Johanna.

**63:49  
Raphaël**Yes, for sure. She did a great job!

**63:52  
Alexey**OK, thanks everyone, and thanks again, Raphaël. Just a reminder, tomorrow we have another podcast episode, so I hope to see you all then.

**64:05  
Raphaël**Hopefully, I’ll see you again soon. Have a good day!

**64:07  
Alexey**Yes, goodbye!

**64:07  
Raphaël**Bye-bye!