



TECHNOLOGY FEATURE Software Engineering / DevOps / Platform Engineering

DevOps Did Not Fail: It Just Split Into XOps.

Understanding the Division of DevOps into Specialized Fields and What It Means for Software Engineers

By **Ibrar Ahmad** | University of Agriculture Faisalabad | Published: September, 2026

✉ ir807573@gmail.com | Position: Software Engineer

Introduction

In 2015, one engineer with a Jenkins job and a Dockerfile was able to handle the whole delivery pipeline by himself. But in 2026 that same pipeline has split into many different fields and every software engineer now needs to pick one to focus on, but he should also know what the others are about. This article explains what happened to DevOps and what actually does it mean for Software Engineers.

Let us think that a kitchen has one chef. He cooks the food, plates it, serves it and washes the dishes, for a small restaurant that works fine. But when the restaurant grows and hundreds of orders come at once then one chef cannot do everything. You need a pasta chef, a grill chef, a pastry chef and one each focused on one thing, each an expert in their own area.

Software delivery in 2026 works exactly the same way.

For a long time one DevOps team took care of everything such as writing the code, testing it, deploying it and keeping it running. That worked when software was simpler, but over time software grew bigger and more complex. Machine learning teams needed their own way of managing models. Security teams needed to check the code automatically at every step. Finance teams also needed to know if how much each deployment was costing.

One team could no more handle all of that. So DevOps did not just fail and disappear, it split into smaller and more focused specialized fields and those specific fields are called XOps which are things like MLOps, FinOps, SecOps, and Platform Engineering. Each one cover a specific part of the job.

This article explains what each specific field does and why AI has actually made things more complicated instead of making them easier and whether you have to learn all of them or just one.

DevOps did not Disappear. It Split into Specialized fields. Those fields are called XOps and understanding them is now a basic requirement for every software engineer working today.



Where It All Started

DevOps came from the problem. Back in the early days developers and operations teams barely talked to each other. Developers wrote the code and passed it on to the operations team who then had to find out how to actually run it and things broke all the time, releases were slow and stressful.

DevOps solved that problem by bringing both teams together and giving them tools like Git, Docker and CI/CD pipelines so that instead of releasing software once a month, teams were able to release updates every day and it improved the work.

But then DevOps teams themselves faced the problem that everything had to be done by them like every deployment, every small change and every pipeline issue. A 2025 study by Harness found that 77% of engineering teams had to regularly wait for another team just to get their normal work done. The system which was supposed to speed the things up had become a problem.

How DevOps Broke Into Pieces

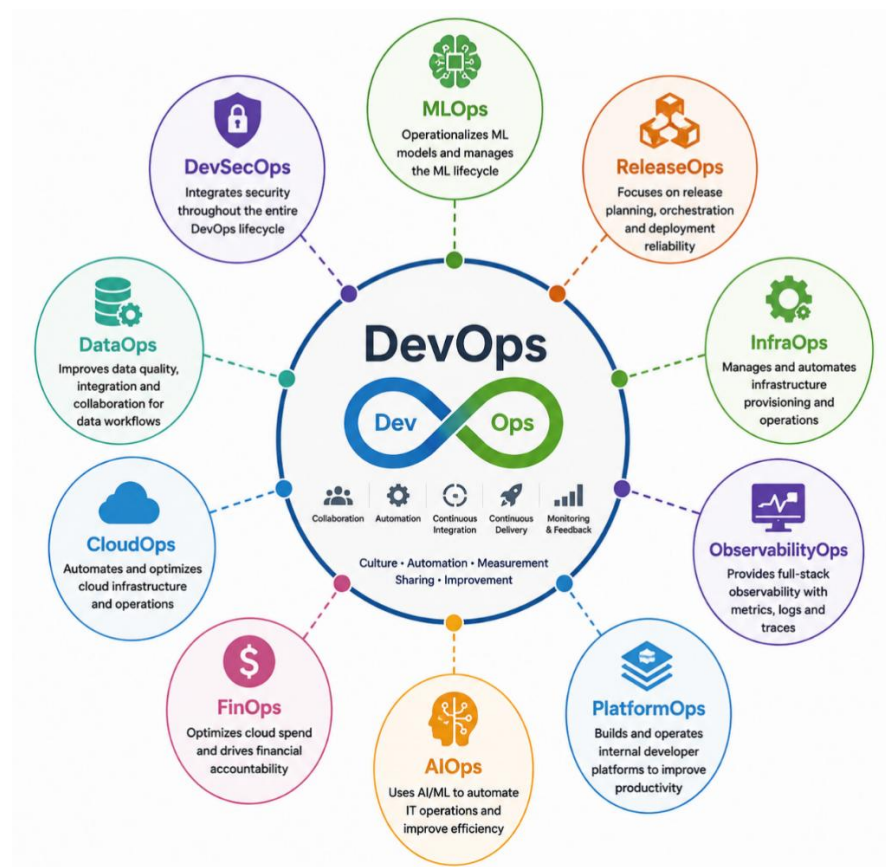
When one approach stops working for everyone, people start building their own versions and that is what happened with DevOps. Machine learning engineers realised that a regular deployment pipeline did not work for their needs. Deploying a model is very different from deploying a web app, you have to track experiments, manage multiple versions of a model, watch how it behaves after deployment and sometimes retrain it from the scratch. So they created MLOps with tools like MLflow, Kubeflow, and SageMaker built specifically for this kind of work.

The security teams had their own problems and security checks used to happen at the very end of development after all the code was already written and where finding the problems that late was expensive and slower to fix. DevSecOps moved that to the beginning and now every time a developer saves their code, he automated tools check it for problems before it ever reaches production.

Platform engineers looked at all this mess and asked a simple question, what if we made it easy enough that any developer could deploy their own code without asking for help? They built internal platforms with ready-made templates and tools so that the developers could work independently. Suppose, Backstage, which is made by Spotify is a well-known example of this.

FinOps came to control the cloud costs. Running AI stuff on cloud servers is expensive and a badly built pipeline can cost thousands of dollars a month, FinOps teams track what everything costs, cut the unnecessary stuff and make sure the company is actually getting profit for what it spends.

The interesting result of all this is that the word 'done' now means something different which depends on who you ask. For a software developer it means the code is written, for a security engineer it means that all the checks are passed and for a FinOps Engineer it means it went out without wasting money and everyone is working on the same product but each of them have different tasks.



Key Findings & Impact

The AI Paradox Nobody Saw Coming

Just when engineers were getting used to XOps, AI tools came and added a new layer of complexity. AI coding assistants became very popular very fast, developers started writing code much faster than before. A 2026 Harness report that surveyed 700 engineers across five countries found out that 45% of developers who use AI heavily are now pushing code to production every single day but for developers who barely use AI that number is only 15%. So yes AI made coding faster. But then the problems started showing up on the other side. The same report found that 70% of heavy AI users ran into deployment failures when they pushed their AI-written code to production, when those failures happened, the average time to fix things was 7.6 hours which is actually longer than it takes for teams that use less AI, on top of that 47% said the manual work after coding which is reviewing, testing and fixing had gotten harder not easier.

The 2025 DORA report found the same thing. About 90% of developers now use AI tools and most say it helps them work faster but the ones using it the most are also the ones hitting the most problems when deploying. This is what people started calling the AI Velocity Paradox. AI helps you write code faster but the pipeline it has to go through with all its security checks, cost rules and deployment steps has not become simpler, you are moving faster but still stuck in the same walls.



AI gives you a speed of a racing car but the XOps landscape is still a road full of speed bumps, you arrive at each checkpoint faster but the checkpoints themselves have not become simpler.

The Fix That Is Already Taking Shape

The good news is that the industry is aware of the problem and people are working on it. At PlatformCon 2025 in New York over 400 engineers showed up in person and 40,000 registered online and the sessions were expected to get 100,000 views total. This kind of interest showed that the platform engineering has become a serious topic, not just something which a very few people talk about. The biggest change at the conference was not a new tool or technology, it was that people finally agreed on how DevOps and platform engineering fits together. Kelsey Hightower said it in his keynote that Platforms are not magic APIs they are agreements that help engineers deliver business value faster.

In clear words DevOps is the shared mindset and way of working. Platform engineering is a practical system which makes that mindset work when your company has hundreds of engineers.

AI is also starting to help reduce the workload on individual engineers. Some companies are now building AI directly into their platforms, tools that can look at a failing deployment and tell you what went wrong or automatically set up cloud infrastructure that already follows all your company's rules.

SO DO YOU NEED TO LEARN ALL OF THEM?

This is probably the most important question in the whole article, so here is a straight answer.

No, you do not need to master all of them, nobody expects that and it is not realistic anyway. What you do need is to pick one area and actually become good at it, if you work with machine learning then focus on MLOps, if you want to build tools that help other developers work faster then go into the Platform Engineering, if you work somewhere where security is critical, a bank, a hospital system or a government app then DevSecOps is where you should go.

But here is what most people get wrong which is choosing one area does not mean completely ignore the others. In a real job your work will be extended to other parts of the pipeline, suppose, if you specialise in MLOps but know nothing about security, your code will get blocked at the security gate every time, if you build platforms but ignore the cloud costs then you will quietly build something that drains the company's budget. Just think about this like you are a heart doctor but heart doctor does not do brain surgery but rather he needs to know enough about the brain to recognise when his patient has a problem there and needs to see a different specialist. You go deep in one field but you have to keep enough knowledge of the others to know when something is outside your field.

For a software engineer the best thing right now is to select one field, learn it properly, get a real experience in it and also spend enough time on the other fields to understand what problem each one is trying to solve and how your work is connected to it.

What's Next?

DevOps splitting into XOps was not a failure but rather it was just what happens when something grows and becomes more complex. Software is now a big part of how every business runs and a single team handling all of this was not going to work forever. The difficult thing now is to make sure that all these separate fields work with each other properly. Platform engineering, better AI tooling and teams



that actually understand each other's work are all working in the right direction. For any software engineer today, the main thing to know is simple. DevOps is not something you learn once and move on from, it keeps changing. The engineers who do well are the ones who select their field of interest and become good at it and stay curious if what is happening in the fields next to them.

About the Author

Ibrar Ahmad is a BS Software Engineering student at the Department of Computer Science, Faculty of Sciences, University of Agriculture Faisalabad. His Email is ir807573@gmail.com.

Keywords: DevOps, XOps, MLOps, FinOps, DevSecOps, Platform Engineering, AI Velocity Paradox, Software Delivery..

References

- [1] Harness (2026). State of DevOps Modernization 2026 Report. Harness.io. Available at: <https://harness.io/resources/state-of-devops>
- [2] DORA / Google Cloud (2025). The State of AI-Assisted Software Development Report. Available at: <https://dora.dev/research/2025>
- [3] Skelton, M. & Pais, M. (2019). Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press.
- [4] Kim, G., Humble, J., Debois, P. & Willis, J. (2016). The DevOps Handbook. IT Revolution Press.