Jacob Achenbach

DevOps is a way of working that brings developers and operations teams together so they can build, test, and release software faster and with more reliability. Before DevOps came around, development teams were moving toward more Agile practices, delivering code in short cycles, meanwhile operations teams were still stuck doing big and risky deployments. This usually made for a lot of frustration because developers wanted to release new features quickly, but operations wanted to keep things stable. DevOps was made to solve this problem by mixing ideas from Lean, Agile, and Continuous Delivery.

Lean thinking actually comes from the Toyota Production System in manufacturing. The idea was to make processes flow more smooth, cut out waste, and improve a little by little every day. When these ideas were applied to software, they meant things like working in smaller batches, spotting bottlenecks, and building quality in earlier stages instead of trying to fix problems later. In DevOps, Lean shows up in methods like value stream mapping, automation, and limiting work in progress so that work can move to production faster.

The Agile Manifesto, written in 2001, added another big influence. Agile focuses on delivering software in short cycles while working closely with customers and being able to change direction quickly. Developers started doing things like continuous integration and automated testing, which made it easier to adapt as they went. But Agile mostly stopped at the “development” stage. Code might be “done,” but it could sit around for weeks before being released. DevOps took Agile’s ideas and stretched them further, making sure operations could keep up with the same fast pace, so working software actually reached users quickly.

The term “DevOps” itself really took off in 2009. That year, Flickr engineers John Allspaw and Paul Hammond gave a talk called “10+ Deploys Per Day,” showing how working together let them release multiple times a day without any problems. Around the same time, Patrick Debois organized the first DevOpsDays in Belgium, and the word stuck. Since then, books like *The*

Phoenix Project and research like *Accelerate* have spread DevOps ideas to more companies and shown with data that these practices really improve performance.

The Continuous Delivery movement tied it all together with the technical practices needed to release software safely at any time. The big push came from Jez Humble and David Farley's book Continuous Delivery in 2010. They put out the idea that software should always be in a deployable state. To get there, teams use automated pipelines to build and test and even deploy their code. Infrastructure is managed as code so environments are consistent, and new release techniques like canary deployments or blue green releases make it safer to push changes out. Continuous Delivery gave DevOps the tools to turn their theory into action.

Technology changes also added to the rise of DevOps. Cloud platforms like AWS made it easy to create and manage servers on demand. Tools like Puppet, Chef, Ansible, and Terraform let teams define infrastructure in code so it was consistent and repeatable. Containers and Kubernetes made deployments more portable and reliable. Microservices encouraged small teams to own their services end to end. Google's Site Reliability Engineering (SRE) introduced the idea of balancing speed with reliability while using error budgets and service level objectives. Recently, DevSecOps brought security into the pipeline, and platform engineering has helped organizations make internal platforms to make DevOps easier for all teams to adopt.

The main part of DevOps isn't just about tools or job titles though it's about breaking down walls between teams, improving collaboration, and making the path from idea to production as smooth as possible. It combines Lean's focus on flow, Agile's flexibility, and Continuous Delivery's automation into a way of working that delivers value quickly and safely. Even though the tools and buzzwords will keep growing, the main goal will stay the same which is to help teams release faster, recover quicker, and keep learning from their systems.

Jacob Achenbach

References

Beck, K., et al. (2001). *Manifesto for Agile Software Development.*

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison-Wesley.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Addison-Wesley.

Allspaw, J., & Hammond, P. (2009). *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr.* O'Reilly Velocity Conference.

Debois, P. (2009). *DevOpsDays Ghent.*

Kim, G., Behr, K., & Spafford, G. (2013). *The Phoenix Project.* IT Revolution Press.

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps.* IT Revolution Press.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. (2016). *Site Reliability Engineering.* O'Reilly.