# Dev-Ops and Systems Management

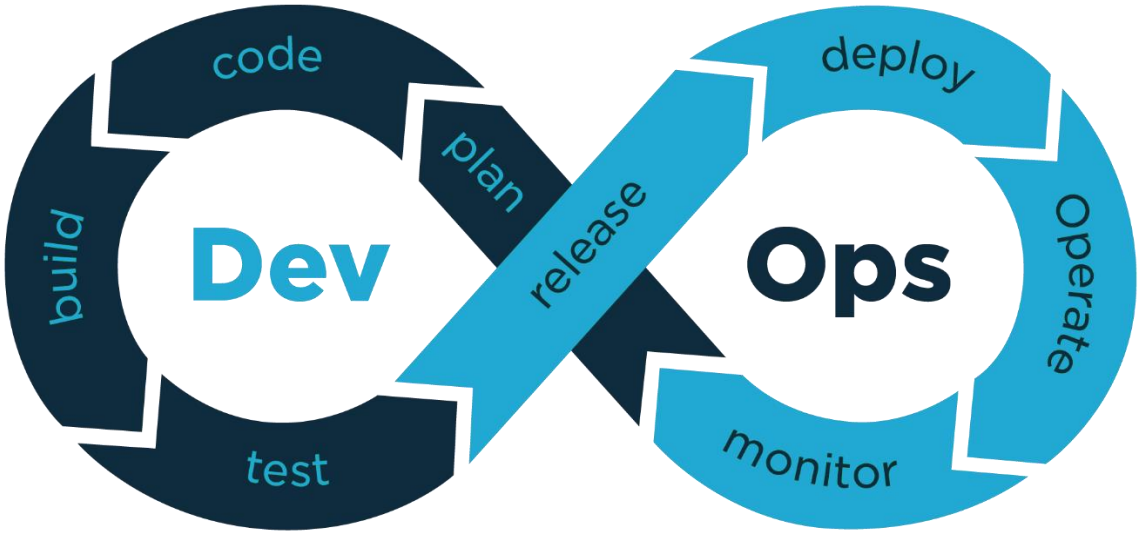
An Exploration of Modern Implementations

Mitch Jean (W0414274)
February 4, 2020
Emerging Technologies (INFT3075)

# Table of Contents

## What is DevOps?

DevOps is at the intersection of software development and IT operations. In essence, it's a framework that defines how software ought to be developed with the support of IT techs to speed up and make the process more efficient, in all ways that efficiency applies. Its core ideal, as you might get from the name, is the collaboration between the development and operations support staff.




In this perpetual diagram, the darker navy-blue sections represent the development team's contribution to a DevOps development cycle, while the lighter peacock-blue represents the responsibilities of operations staff. This dictates that the code written by developers is tested internally, the release build is passed off to operations staff, the build is released, operated, then the performance and integrity is monitored/supported over a period of time. In the context of this example, systems management would obviously sit on the operations side. If I had to make an educated guess, I would say that the "monitor" portion is the biggest responsibility in systems within DevOps. Seeing as the emphasis is on getting the product out the door as efficiently as possible, it's up to systems managers to ensure that the deployed code is running well and doing what it's intended to do, especially if corners were cut to ship the code quickly. All of the other steps are just as vital, but the job of systems is typically to maintain efficiency over an extended period.

This synergy between these two distinct teams defines what problems a DevOps environment can solve. Taking a traditional non-DevOps approach, the development team is often unaware of challenges faced by operations staff in order to get their application/program running. A DevOps approach tries to bridge this gap by having both sides communicate more effectively and frequently, as well as implementing measures during development based on the suggestions of operations staff in order to make deployment and monitoring easier and more effective. Operations/QA staff can also be left in the dark as far as the business logic/value behind certain decisions and features in development, which can lead to conflicts and inefficiency within the team, especially when something goes wrong with one or the other.

## DevOps vs. Agile Development

As mentioned before, DevOps tries to solve for discrepancies between development teams and IT operations teams, which remedies intra-team issues and deployment/development complications. Agile, on the other hand, is concerned with bridging the gap between the development team and stakeholders/customers, especially surrounding their expectations for the end product. This is to say that the two frameworks work to solve different intrinsic problems, but in a very similar way. These two concepts are not mutually exclusive, but they can sometimes be at odds if prioritizing workflow and automation gets in the way of iterative releases and consistent communication with the customer (DevOps being the former, Agile the latter). Some even say that DevOps was built on the foundation that Agile laid, that being a generally accepted framework of best practices for software development. Both are IT industry staples in the modern day, but Agile was certainly ahead of DevOps as far as being a widely adopted, even "trendy" methodology for software development, and it would have made sense for the pioneers of DevOps to take pointers from Agile as they were developing their model.

The biggest difference to me is in the implementation of the two methodologies. More than anything, DevOps is a concept of culture where there aren't as many established practices - it simply defines how two teams should synergize and consider one another. Agile, on the other hand, employs many "hard coded" practices such as scrum meetings, Kanban, XP, etc. This isn't to say that these sorts of concepts don't exist in a DevOps environment, they're just not ingrained into the framework.

## DevOps Challenges and Considerations

The biggest challenge for a systems admin surrounding DevOps would be acclimating to an environment that demands it. A company whose strategy employs DevOps needs its employees to be on the same page, at the very least on a technical level. Any systems-oriented IT professionals will also need to understand the fundamentals of the code being written by their developer counterparts. This doesn't begin and end at the syntax of whatever language they're writing in, either; DevOps pros need to understand how to optimize a dev environment and deployment scenarios for the project at hand. As mentioned before, this demands knowledge of the programming languages being used (at least to a degree), and how to build/maintain infrastructure around it. It's vital for systems managers to keep all dependencies are present, all updates are installed (after they're deployed to a test environment, of course), and to make sure everything maintains stability into the future.

Security considerations when implementing DevOps may seem straightforward and obvious, but since there are so many more moving parts to a typical DevOps environment than say, a Windows active directory with a few printers, some are worth mentioning. There may be containers that need to be hardened, web applications on potentially vulnerable servers, channels of communication that need to be secure, just to name a few. The most important security factor for systems administrators in DevOps roles is to stay up to date on every "piece of the puzzle" used. As a random example, if a patch comes out for Apache and Python, a DevOps admin will need to determine what impact this will have on the stability of the system(s), as well as if any new holes in security will need to be patched up. Once again, if the DevOps team is practicing what they preach, any and all of these changes would be tested before going to a live environment.

# References

Oteyowo, T. (2018, April 12) DevOps in a Scaling Environment. Retrieved from https://medium.com/tech-tajawal/devops-in-a-scaling-environment-9d5416ecb928

Medium (2018, April 12) DevOps Cycle Diagram (Graphic). Retrieved from https://miro.medium.com/max/1982/1*AwvDJDfErlD34ox2QpwGoA.png

Guru99 (n.d.) Agile Vs. DevOps: What's the difference? Retrieved from https://www.guru99.com/agile-vs-devops.html

CollabNet (n.d.) What is DevOps? The Ultimate Guide to DevOps. Retrieved from https://resources.collab.net/devops-101/what-is-devops

Brodie, S. (n.d.) From agile to DevOps to continuous delivery: An evolution in software delivery. Retrieved from https://techbeacon.com/app-dev-testing/agile-devops-continuous-delivery-evolution-software-delivery