# DEVOPS CULTURE: WHAT IT MEANS, WHY IT MATTERS

DevOps is an SDLC methodology designed to expedite product release cycle, reduce waste process and improve end-user experience with every release iteration. In addition to the tooling and processes that enable DevOps, the popular SDLC methodology is largely focused on culture, the people and their mindset.

In a DevOps organization, small, multidisciplinary and autonomous teams work collaboratively across the Dev, QA and Ops departments with the shared responsibilities. DevOps culture spreads across teams working on development projects as well as the organization as a whole. Teams are required to focus on product quality and speed of delivery through collaborative efforts, automation and response to feedback from all stakeholders. The organization is required to eliminate the silos across departments and allowing teams to operate autonomously, while adopting governance measures and policies that facilitate autonomy and automation-driven SDLC processes.

DevOps promises a system of continuous improvement through collaborative efforts of several contributing teams and individuals throughout the SDLC pipeline. Despite the available technologies and process workflows in place, teams may face conflicting goals and may not be able to realize the true pace of DevOps if a cultural change is not adopted. For instance, Devs may want to deploy new features quickly in response to end-user feedback and market demands, whereas QA may want to ensure that every release iteration is stable, secure and performs as per intended standards. With traditional SDLC models, this deadlock emerges because Devs inherently regard security as a responsibility of QA, who are inclined more toward keeping systems running instead of releasing fancy new features into the market. With the DevOps mindset, Devs take early responsibility to check in functioning code with every build iteration and working with the QA to fix potential bugs. And to address the issues associated with performance and failures, DevOps teams focus on

reducing the Mean Time to Mitigate/Remediate (MTTM/MTTR) instead of optimizing the Mean Time Between Failure (MTBF). The former strategy allows DevOps teams to deploy multiple feature releases fast, be prepared and react fast to reduce the impact of potential failures, whereas the latter only aims to reduce potential issues. In reality however, software issues are rarely eliminated altogether and extra efforts toward achieving this goal compromise the pace of release cycle.

The Collaboration part of the DevOps culture is also different from traditional SDLC models. While communication across otherwise siloed teams and departments as necessary is highly encouraged, DevOps aims to reduce bottlenecks within collaborating teams through effective automation. DevOps aims to remove requirements for manual communication, approvals and processing delays between team members. For instance, when Devs need to additional hardware resources in a DevOps set up, they should be able to deploy their builds in cloud environments without having to interact with Ops or wait for their approval. Automation further extends the ability of DevOps members to collaborate. Automated tests and server configuration allow Devs, Ops and QA to understand the performance of every build iteration, how the servers are configured, reducing the possibility of human errors and delays caused due to manual interactions.

Developing a true DevOps culture at the workplace requires teams to redefine responsibilities, adopt proactive measures to reduce problems rising due to the workplace culture and adapting cultural performance on a continuous retrospective analysis.

Assignment of roles and responsibilities may not end with the first meeting. Know that teams tend to follow multiple phases of growth, and not necessarily in the forward direction with new job tasks and projects coming into the pipeline. Following Tuckman's Group development process, teams may not remain in the Performing state after following the Forming, Storming, Norming stages. The DevOps requirement of continuous improvement based on real-world feedback may cause individuals to detract from their original roles and responsibilities. As a result, an ongoing refinement and consolidation of team responsibilities may be required.

DevOps culture blurs the lines between the roles of Devs, Ops and QA. Instead of following the traditional procedure of handovers and sign-offs, team members work collectively across the SDLC lifecycle and adopt a shared responsibility for the success and failure of their software projects. Instead of indulging in the blame-game, Devs, Ops and QA work together from the beginning to deliver working software builds through every iteration. From the perspective of the organization, traditional decision-making mechanism and governance policies are also redefined for the DevOps environment to facilitate this collaboration. For instance, instead of following manual reviews and sign-offs, teams can rely on version control systems with automated approval workflows. Such measures allow teams to operate autonomously while also managing risk and fear of failure. Members from previously siloed departments are therefore able to step out of their comfort zone and collaborate with an open mind. Devs, Ops and QA learn to embrace the change and adopt shared responsibilities instead of identifying each other as members of separate teams, albeit working on the same SDLC project pipeline.

The philosophy of continuous improvement with DevOps is exercised by working toward frequent but small software changes with every build iteration, carefully tested with automated tools and released to end-users at a faster pace. Response from end-users and stakeholders allow developers to adjust the SDLC pipeline continuously for improved outcomes. These improvements are inherently customer-driven and focus on improving end-user experience with every release.

In conclusion, implementing DevOps requires a significant change in the way individuals work with

each other and how organizations facilitate the necessary cultural shift. A preliminary starting point is the mindset: be transparent and trust each other; adopt non-conflicting goals; embrace failures instead of indulging in blame-games; and follow a shared sense of responsibility instead of the 'not my job' philosophy. From a process and organizational standpoint, enable true autonomy among individuals and DevOps team members; facilitate cross-functional collaboration; reduce waste and bottleneck process; and adopt continuous flows across the SDLC pipeline, including integration, testing, deployment and even funding, among others. Even though a sudden cultural shift is not possible for most organizations, starting off small and expanding the cultural approach across multiple DevOps teams is a viable starting point.