DEVOPS CULTURE & HOW TO SUCCESSFULLY ACHIEVE DEVOPS

DevOps is a software development lifecycle (SDLC) methodology designed to:

- Expedite product release cycles
- Reduce waste processes
- Improve the 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. So, what is this DevOps culture everyone talks about? Let's take a look.

(This article is part of our DevOps Guide. Use the right-hand menu to navigate.)

What is DevOps culture?

In any DevOps organization, whether it's a single team or the entire organization, small, multidisciplinary and autonomous teams work collaboratively across the developer, QA, and Ops departments with shared responsibilities. Teams are required to focus on product quality and speed of delivery through:

- Collaborative efforts
- Automation
- Response to feedback from all stakeholders

The organization itself is also required to:

- Eliminate silos across departments

- Allow teams to operate autonomously
- Adopt 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.

(Explore the principles and values of DevOps.)

DevOps requires cultural change

Despite the available technologies and process workflows in place, if you fail to adopt the cultural changes of DevOps, teams will likely face conflicting goals and may not be able to realize the true pace and promise of DevOps.

For instance, Devs may want to deploy new features quickly in response to end-user feedback and market demands—but the QA team may want to ensure that every release iteration is stable, secure, and performs as per intended standards.

With traditional (waterfall) SDLC models, a deadlock emerges: 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, however, 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 Resolve (MTTR) instead of optimizing the Mean Time Between Failure (MTBF).

Let's compare strategies:

- The former strategy allows DevOps teams to deploy multiple feature releases fast, be prepared and react quickly to reduce the impact of potential failures.
- The latter aims only to reduce potential issues. In reality, however, software issues are rarely eliminated altogether. Extra efforts toward achieving this goal compromise the pace of release cycle.

Collaboration & automation in DevOps

The collaboration part of the DevOps culture is also different from traditional SDLC models.

Communication across otherwise siloed teams and departments as necessary is highly encouraged, but doesn't end there. DevOps aims to reduce bottlenecks within collaborating teams through effective automation. DevOps aims to remove requirements for:

- Manual communication
- Manual approvals
- Processing delays between team members and/or other staff

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 the necessary 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, and how the servers are configured, in order to reduce:

- The possibility of human errors
- Delays caused due to manual interactions

(Learn more about automation in DevOps.)

Building a DevOps culture

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 follow the Forming, Storming, Norming stages but still not reach the Performing stage. 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. 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 in your work
- Trust each other
- Adopt non-conflicting goals
- Embrace failures instead of indulging in blame-games
- Develop a shared sense of responsibility instead of the 'not my job' philosophy.

From a process and organizational standpoint, you'll need to:

- Enable true autonomy among individuals and DevOps team members
- Facilitate cross-functional collaboration
- Feduce waste and bottleneck process
- 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.

(Understand change theory to better adopt these changes.)

Defining successful DevOps

Here are some key signs that you've achieved a DevOps environment:

- 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.
- The organization redefines traditional decision-making mechanisms and governance policies 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.

DevOps supports agility, continuous improvement

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.

Related reading

- BMC DevOps Blog
- When Cultures Clash: Removing Friction Between Dev and Ops
- Book Review of The Phoenix Project: DevOps For Everyone
- Top DevOps Trends Today
- The State of DevOps
- DevOps Transformation: Metrics That Show Business Value