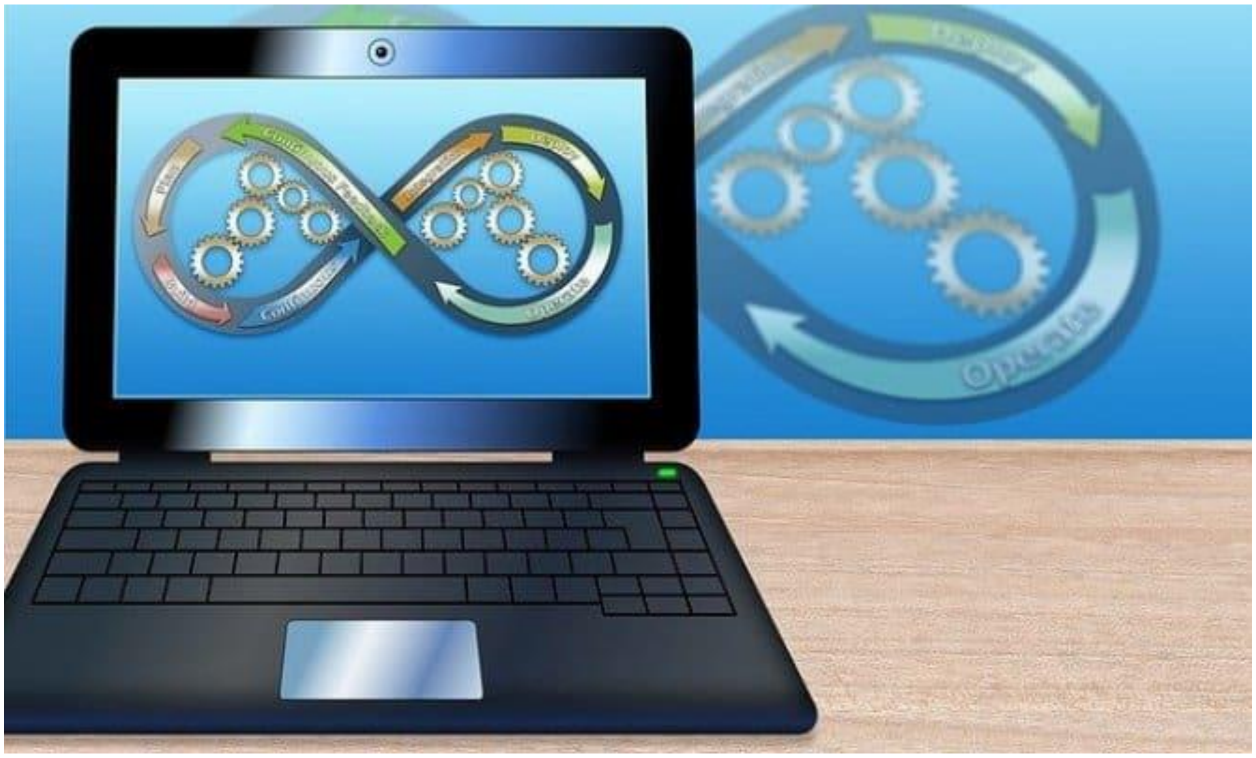
DATAQUEST

Software AUTHOR: DQINDIA ONLINE - NOVEMBER 2, 2020



The Practice of DevOps

Software development is both a science and an art. The science involves models, methods, and procedures that have been shown to be useful and correct. The art is weaving together these methods for a given situation, a given problem, a given set of programmers, and a given client, to deliver on the project's promises. Thus the software industry has evolved with twin personas – the formal models side and the actual practices side. There are those who adhere strictly and firmly to formal models, while most want to find a variation that suits their situation and their needs.

DevOps is a new model of building and implementing software, though many will argue it is a "mindset" more than a model. The DevOps approach has grown in popularity in recent years, building on the Agile approach, which is the dominant method of constructing software. DevOps has grown in popularity owing to the massive growth in computing needs, which has arisen from the use of billions of mobile phones, and their dependence on Cloud-based applications. Further, there is now massive competition for apps on various platforms, and software developers feel compelled to roll out applications, their enhancements and features on a frequent and continuous basis.

The question we address is how closely do firms and software engineers follow the formal model of DevOps and how far do they adapt and change the model. We interviewed software developers, DevOps engineers and product managers to understand the industry practices related to DevOps. Our findings are based on specifics of the process, which we outline below.

Continuous Development
We found multiple interpretations for this term. Some consider continuous development as an umbrella term to define the full software development cycle which includes integration, deployment, and testing. Others don't consider this as a part of DevOps at all, but as a part of the Agile approach, since they relate it to planning and development.

Continuous development of software was largely avoided in the earlier, Waterfall model of software development, where the scope of the project was defined early in the process, and frozen after that. The Agile approach changed this by saying that the velocity of development, which is the rate at which development will proceed, is fixed, but the scope remains flexible. This does lead to scope creep where the limits of the project, or the point at which it is considered complete, are unclear. This is welcome in DevOps, where the only constraint is that scope is not changed in the middle of a development cycle, a Sprint, but after it is over.

Continuous Integration
Continuous integration (CI) is the process that aligns the code and build phases in the DevOps pipeline. This is the process where new code is merged with the existing structure, and engineers ensure that everything is working fine. The developers who make frequent changes to the code, update the same on the shared central code repository. The repository starts with a master branch, which is a long-term stable branch. For every new feature, a new branch is created, and the developer regularly (daily) commits his code to this branch. After the development for a feature is complete, a pull request is created to the release branch. Similarly, a pull request is created to the master branch and the code is merged.

We have seen slight variations to these practices across organisations. Sometimes the developers maintain a fork, or copy, of the central repository. This limits the merge issues to their own fork and isolates the central repository from the risk of corruption. Sometimes, the new branches don't branch out from the feature branch but from the release or master branch.

Small and mid-size companies often use open sites like GitHub for their code repository, while larger firms use Bitbucket as their code repository, which is not free. Bitbucket is more flexible and the continuous integration and delivery get tied up with the source code from the start. This saves setup time and less management is needed for repositories and servers. In contrast, for GitHub, everything needs to be setup on a case-by-case basis.

Continuous Testing
This is the area where we found there was a maximum deviation. Some of the major concerns were

- Lack of dedicated quality assurance (QA) testers in the team.
- Developers found writing the test cases to be boring and neglected it when given a choice.
- Lack of set processes in the organization that mandated test cases.
- Covering all the unit test cases, automation tests, was time-consuming and was considered a waste of time.
- Lack of testing on old features developed by predecessors. Extra time was needed for the developers to understand the code and write tests for them.

Regardless of the concerns, we found improvement in testing in all organizations over time. Test coverage has improved, and organizations are shifting focus to incorporate continuous testing to reduce the cost of errors by trying to catch the bugs early. If a product has continuous testing as part of its process, the chances of encountering a major bug after the feature release is significantly reduced.

Continuous Delivery
Continuous Delivery (CD) is the process by which changes of all types – new features, configuration, and bug fixes are rapidly delivered to the end-users. Usually, teams face problems in achieving a completely automated CD pipeline. Major reasons cited are:

- Different technologies and tools involved (like containers and scripting) that are difficult to integrate.
- Lack of dedicated DevOps engineers with the required know-how.
- Variation in tools depending on the codebase. For example, Visual Studio supports code written in C# and .NET frameworks, but Java requires Jenkins.

To overcome these problems, teams resort to their customized solutions. This leads to semi-automated solutions which differ from team to team even within an organization. Some teams prefer to deploy to multiple environments (via tools like Ansible), whereas others write Python scripts for doing the same thing, based on their skill sets.

Moreover, it is practically impossible to completely automate the continuous integration and delivery pipeline as the number of development environments increase. For some firms, there can be around 15-

20 different environments, depending upon the product. In each environment, testing and sign-off needs to be done by different stakeholders. Hence, even for products that have an automated CI/CD pipeline we typically observe the process to be automated till the QA part, after which the deployment is manually triggered. Although the process is mostly seamless and doesn't require more than a few clicks, but manual intervention is still required.

In conclusion, we found that DevOps is a growing practise and firms are finding ways to manage their projects in as smooth a manner as feasible. There is considerable demand on software developers to learn new tools, which requires some effort, but the payoff is smoother operations and better quality.

By Rahul De' – Professor of Information Systems, TirthaTushar Panda and ShreyashTade – MBA students, IIM Bangalore