DevOps — 7 Best Practices for Implementation

An extension to the preferred agile model, DevOps' adoption has been growing faster than ever, gaining tremendous fame in the past few years. Keeping communication and integration as the key, it leverages on collaboration between operations, development and businesses, thus ensuring seamless and efficient deliveries.
The increasing number of organizations leveraging on DevOps to unleash the untapped potential and close barriers amidst their departments to build quicker and robust software and execute a seamless delivery shows the potential DevOps has.

Proven Productivity Increase

Data proves that organizations following DevOps have seen a business expansion by 38%, as well as a productivity increase by 51%. However, despite understanding DevOps and its working modalities, many companies struggle to efficiently incorporate DevOps into their IT framework.

Perhaps because this is a mindset and cultural change to be adopted rather than as a checklist for mere actions.
Listed below are some DevOps best practices that can certainly help.

1. Automation
2. Integrated Configuration Management

3. Integrated Change Management
4. Continuous Integration
5. Continuous Testing
6. Continuous Delivery
7. Security — Monitoring & Alerts

Automation

The implementation of DevOps depends a lot on automation; for configurations, infrastructure set-up development and testing, and deployment — to ensure more frequent yet quality delivery is a possibility.

For instance: early identification of errors in development gives more opportunities for them to be rectified while in the development phase itself. Hence, taking into account the continuous operations in parallel with ongoing development, a nimble automated testing framework adds more fuel to the SDLC. Test automation can simply be achieved by determining certain test cases, then running and analyzing their relevance in different scenarios, and achieving quicker, consistent, accurate deliveries.

The automation starts at the code generation level, continues till the code has moved to production, and even post it to monitor the application. The end-to-end DevOps pipeline from CI, CD, CT and performance of the application is automated.

Integrated Configuration Management

As the essential part of operations, configuration management enables agility — the base to DevOps.
Optimizing the use of existing systems in place and maintaining system-wide configurations across networks, servers, application, storage, and other managed services, configuration management enables the development teams to look at the bigger picture.

It promotes the utilization of the existing services during the software development rather than investing time and efforts in reinventing the new services from scratch.

Integrated Change Management

With technology constantly evolving, it is essential for organizations to be agile with change management and yet deliver on time with optimal quality. Hence defining processes that are required to be followed to cater to these changes, and the methods and techniques for execution — all play an equally important role.

The DevOps principles to expedite faster releases breaks down these changes into smaller sets and automates their management.

Overall the organization needs to have an open mindset and defined processes for any such ‘changes’ and that can be done by defining the scope of change, updating systems with the new scope and including all teams to relook at the impacts–hence communication and collaboration that are the base for DevOps.

Continuous Integration

To expedite releases, organizations have developers working on independent features parallelly which are later to be clubbed. With each one making changes to their own set of codes, without knowing the impact it may create to the other set of features, the ‘merge day’s as the integration days are called, are mostly full of errors, resulting in tedious and time-intensive.

Even when it comes to rollbacks, CI is helpful. However, a roll-back strategy needs to be thought through well as it differs from application to application.

Continuous Integration practices allow developers to carry out integrations sooner and frequently and with the help of CI tools, simplifying the integration challenges by automating code tests to ensure they aren’t broken and will integrate seamlessly. Hence continuous testing is important for both CI and CD.

Thus ensuring the development of high-quality software by providing regular and immediate feedback, and fixing them. Read for more about this blog :DevOps — 7 Best Practices for Implementation

#DevopsContinuousTesting #DevOps #DevopsTestingServices #DevopsTesting