While automation has become more involved in many aspects of our lives, build automation is no different when it comes to DevOps.

Build automation for DevOps is about automating the process for preparing the code for deployment. This deployment is done usually to a live environment depending on the framework and coding environment. Depending on the programming language or the environment configurations to create the code may require specific requirements to be carried out before it's ready for deployment.

Some of these requirements are for the code to be compiled or run certain quality or case tests with build automation taking care of these steps. Build automation has the great advantage of saving time on elementary and uninteresting tasks while being consistent on its delivery.

Before building automation, each developer would use their own integrated development environment while working on the source code. They would then be required to compile or build the code, which created an artefact that could be passed on for deployment. Using individual IDE's could create version control issues, especially in large teams .

Utilising build automation to do these basic tasks while ensuring version control and automatically deploying the latest code to keep everyone in the team on the same page. Solid build automation happens independently of an IDE. Even if it is possible to execute the build automation within the IDE, it should also work the same way outside of an IDE, which is essential because it enables us to execute builds on a continuous integration server.

Another important thing about good build automation is that it should be agnostic of the machine's configuration that it is built on. That means that if you can build the code on your machine, I can also build the code on my machine, and if we use a continuous integration server, we can build the code there and get the same result in all three places.

So why should you do build automation? First, build automation is fast. It automates tasks that otherwise might have to be done manually, which saves time. Secondly and even more importantly, build automation is consistent because the build happens the same way every time. This eliminates the human error and confusion that arise from manual builds that might be done a little differently every single time.

Build automation is also repeatable, which means I can take the same version out of source control and build it multiple times and get the same result every time. Any particular version of the source code can be transformed into deployable code in a consistent way. Build automation is also portable because the build can be done the same way on any machine.

Any team member can build the code on their machine and build the code on a shared build server like a continuous integration server. Building code does not depend on specific individuals, and it does not depend on specific machines.

To sum up, build automation is a developer-centric process. Build automation makes building code more reliable because it's automated at key stages. This will ensure there will be fewer problems that are caused by poorly done manual builds.