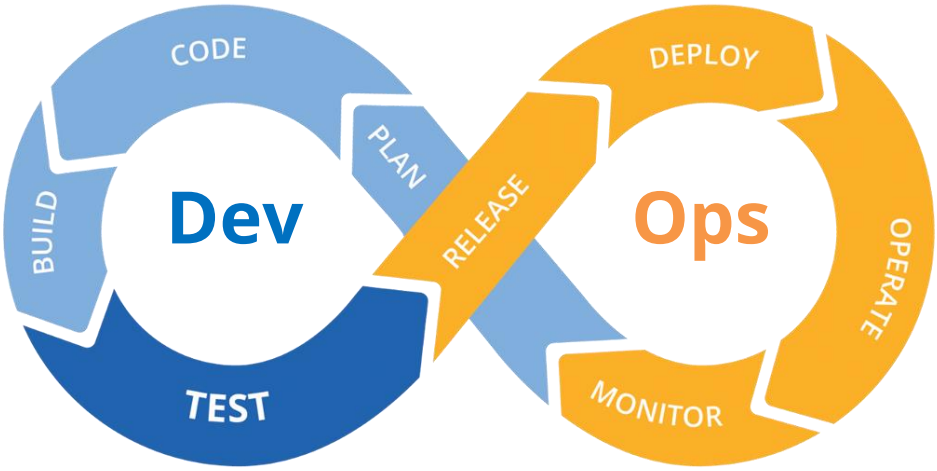
# DevOps with GeneXus

## Introduction

GeneXus™

DevOps is methodology where the creation of an app is integrated with the operations that follow the development, such as production cutover, maintenance and software evolution, in an endless cycle. Let us now see how this applies to GeneXus.

## DevOps definition



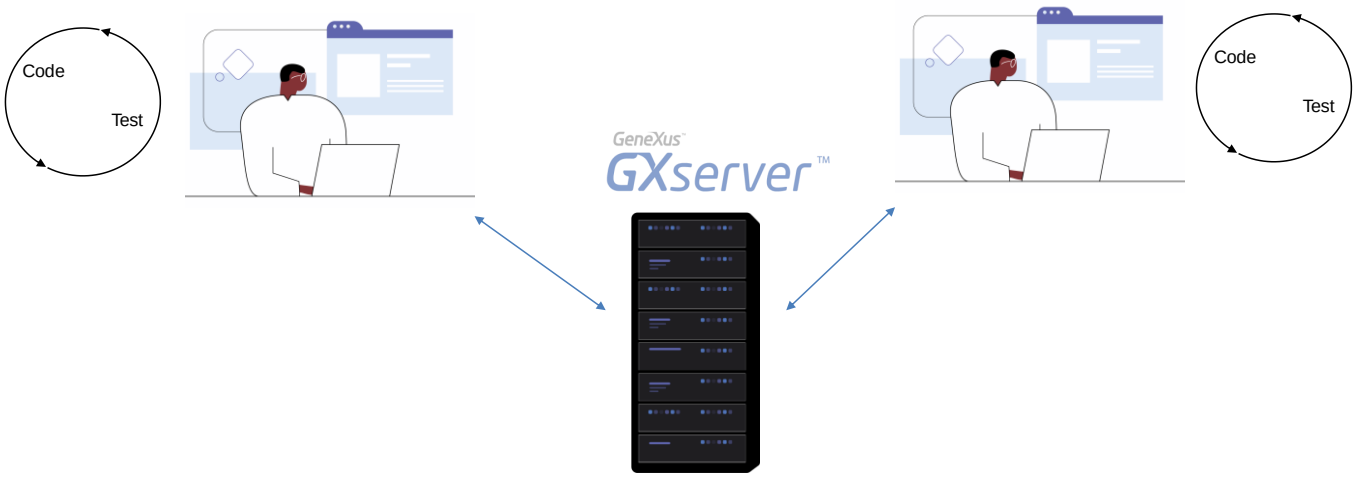

DevOps comprises a group of agile practices that combines software development "Dev" with information technology operations "Ops" in order to shorten the life cycle of system development while offering software in a continued manner.

The "Dev" (development) cycle includes the application's design, development, coding, building and testing; and the Ops (operation) includes its release, production cutover, operation, and monitoring. This monitoring will determine future changes required in the software, such as maintenance due to flaws fixed, or the evolution implied in the addition of new functionalities that enable a better use of the app or keep it fully valid.

Integrating all tasks within a continuous cycle enables companies in the field of system development to benefit from market opportunities faster and also to reduce the time necessary to include customer answers, thus enhancing their experience with the software they use.

Let's now see the main problems that come up in an app's life cycle.

## Integrating changes




Save for exceptional cases, software development is always a team activity; and one of the main problems to solve there is the integration of the changes that each developer makes to the app.

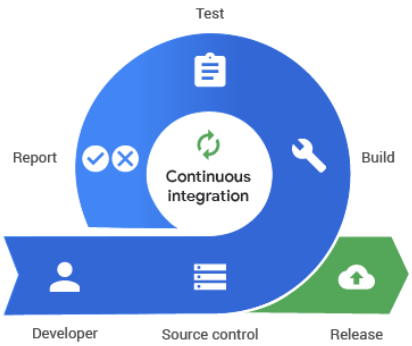
Every developer works on developing a functionality, within a cycle where encoding and testing is done until the objective sought is achieved.

But, what happens when we want to integrate the work of developers so that the functionalities they worked on are included in the app's following release?

Each of them will try to upload their changes to GeneXus Server, but probably many conflicts will occur, particularly if the developers worked independently for a long period of time. They may have defined objects with the same name but for different purposes; or maybe one of them modified existing objects by removing functionalities on which other developers are relying.

## Continuous Integration

- Frequent integration of changes, at least once daily.
- Integration verified automatically.




The idea about continuous integration implies the frequent integration of work, more than once daily, if possible, so as to avoid conflicts, or to just have small conflicts that are easy to fix.

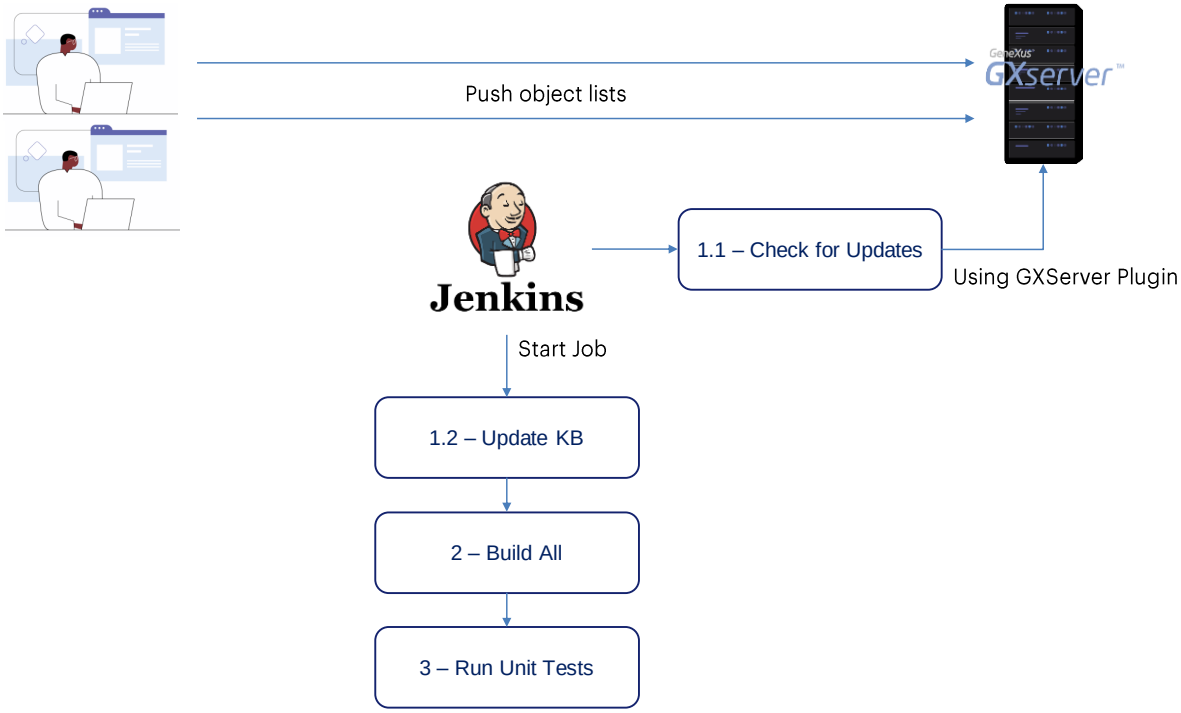
After the integration is produced, the app must be set up and tested automatically to detect integration errors as soon as possible. Automatic testing is also useful to make sure that nothing already in function has stopped functioning (known as regression testing).

## Continuous integration tools

Jenkins

GeneXus GXserver™

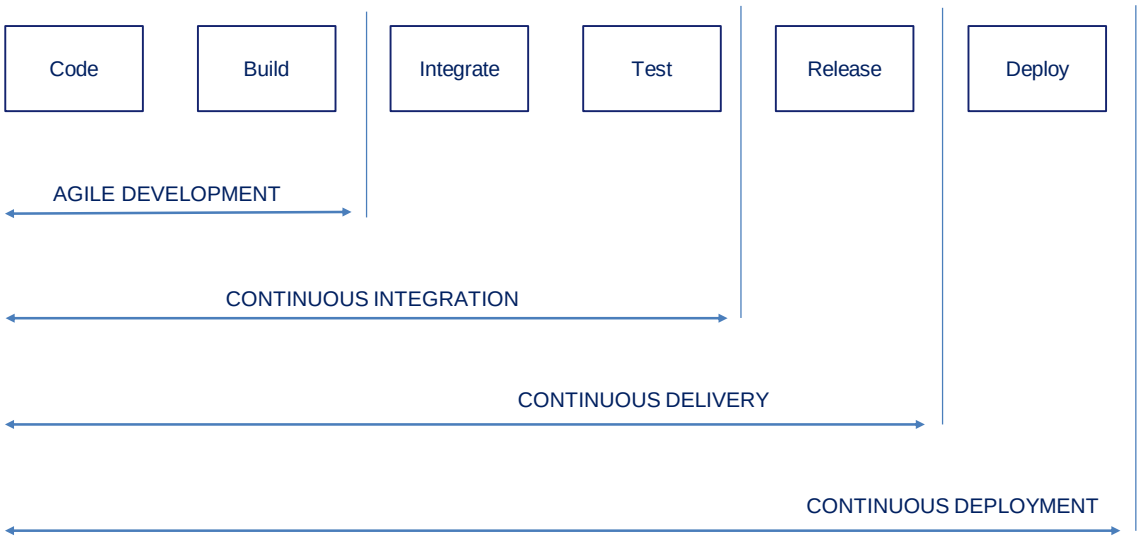
To this end, we will be using GeneXus Server along with Jenkins, a continuous integration engine. This tool will implement the whole process.




For each Commit by a developer, at least the following automatic process will be carried out: Update of KB, Build All, and execute tests.

When everything goes well, the Build is successful and we may move on. Otherwise, the process must be stopped to solve the conflict encountered.

## Stages in an app's life cycle




These steps make up the life cycle in an application.

Our development method may be considered agile when our process is capable of speeding up the development and building process.

When we add automated integration and validation to our process we have achieved Continuous Integration.

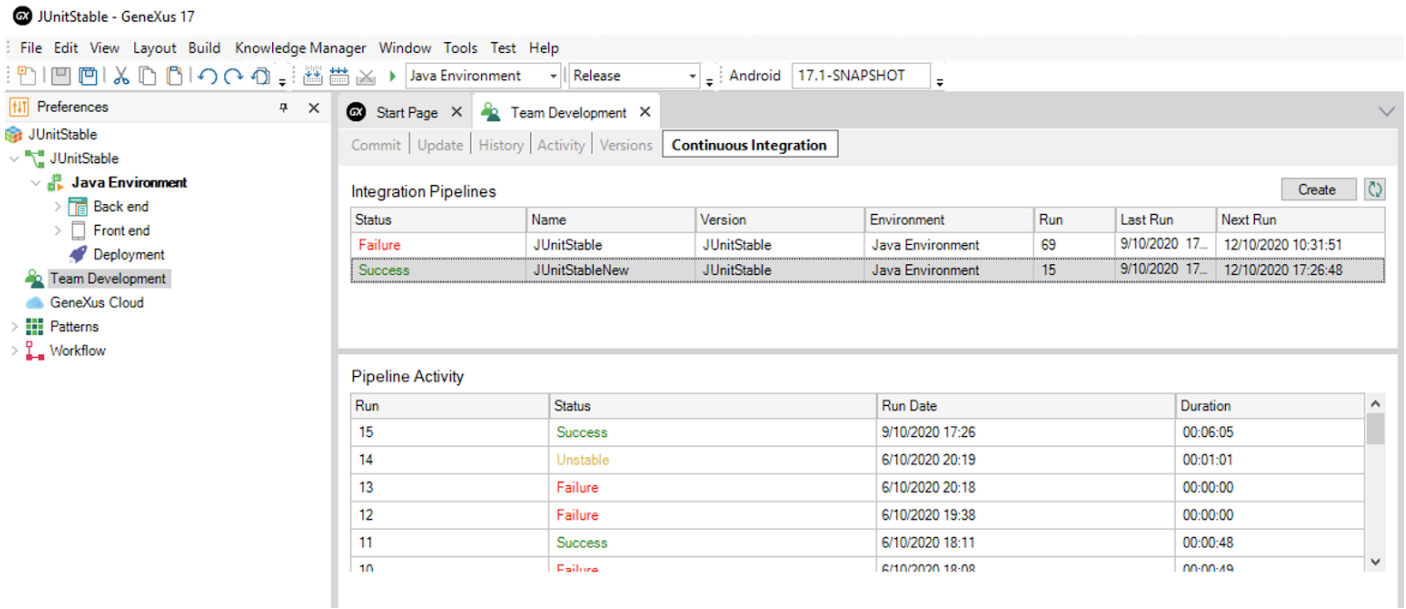
We have attained Continuous Delivery when we also add automated release setup, so that we can move on to publish to production with just one click.

If our process is so smooth and automated that we may reach automatic production cutover, then we have Continuous Deployment.

## DevOps Process

And when our process also includes the app's operation and monitoring in production –so that information is generated as feedback for planning which functionalities should be developed from there– we can finally say that we have a DevOps process.

## Creation and monitoring of Continuous Integration processes




As we've seen, in order to achieve the DevOps methodology, we must make sure we achieve Continuous Integration; that is, after developing the application, automate the integration of changes and validation.

It is possible to create and monitor continuous integration processes (pipelines) from the GeneXus IDE and the GeneXus Server console.

Once we upload the KB to GXserver, if we go to Knowledge Manager / TeamDevelopment we find the "Continuous Integration" tab where we can create pipelines, view when they were executed, the commits made in each execution, their result, and the complete log of each execution.

More information at:
https://wiki.genexus.com/commwiki/servlet/wiki?40706

For more information on the use of the DevOps methodology in GeneXus, visit this Wiki link:
https://wiki.genexus.com/commwiki/servlet/wiki?40706

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications