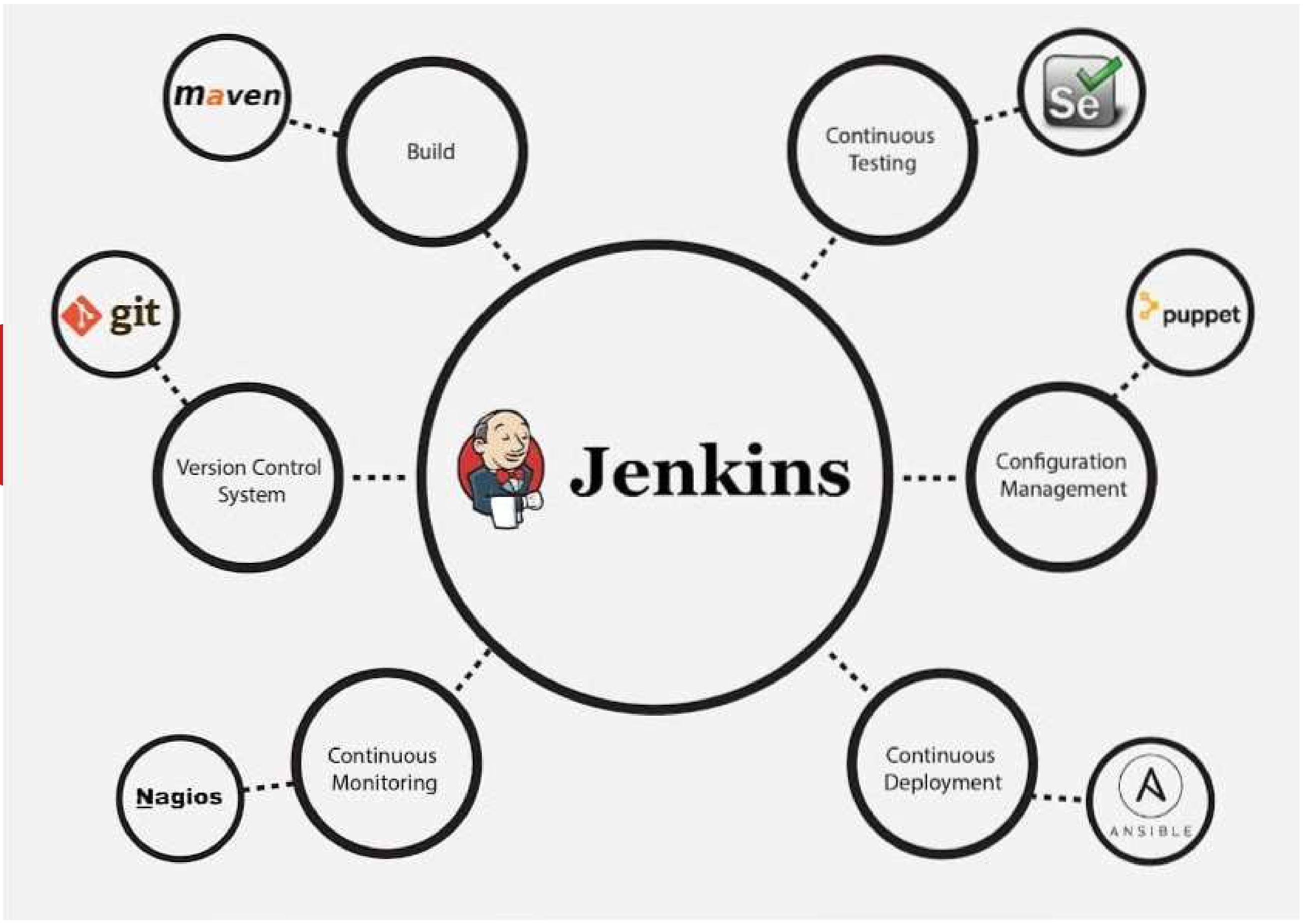
Jenkins


maven
Build
Continuous Testing
Se
git
puppet
Version Control System
Jenkins
Configuration Management
Continuous Monitoring
Nagios
Continuous Deployment
ANSIBLE

Problems if we will not use CI

Developer teams have to wait till the complete software is developed for the test results.

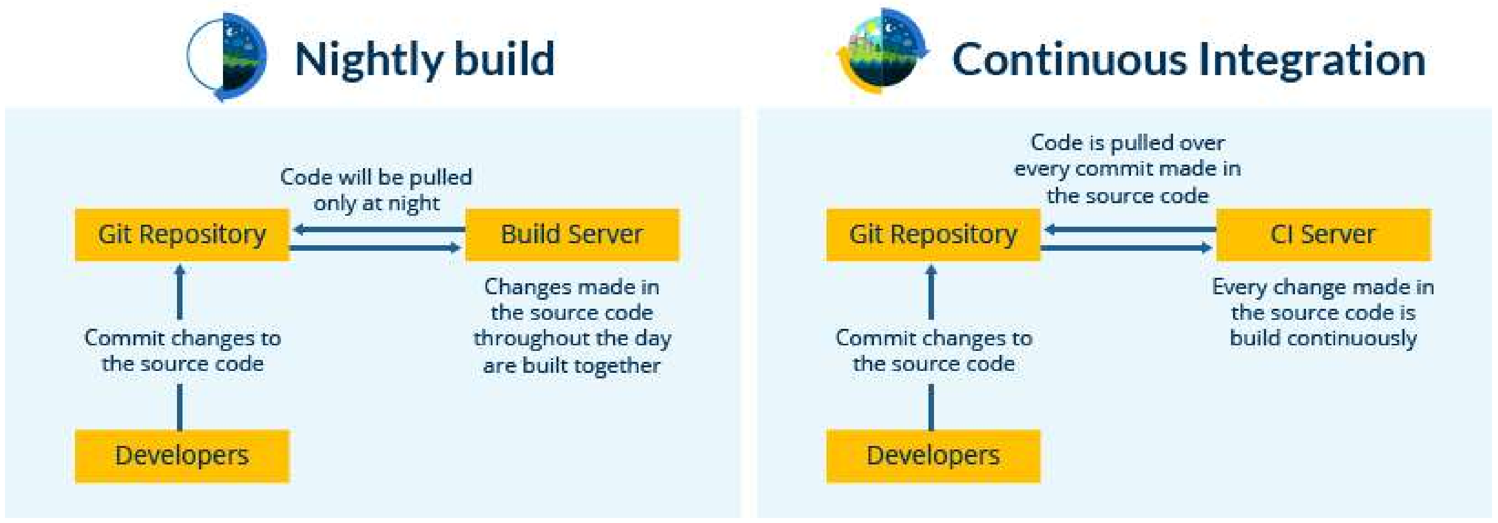
There is a high prospect that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.

It slows the software delivery process.

Continuous feedback pertaining to things like architectural or coding issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.

The whole process was manual which increases the threat of frequent failure

Continuous Integration With Jenkins


Nightly build
Code will be pulled only at night
Git Repository
Build Server
Commit changes to the source code
Changes made in the source code throughout the day are built together
Developers
Continuous Integration
Code is pulled over every commit made in the source code
Git Repository
CI Server
Commit changes to the source code
Every change made in the source code is build continuously
Developers

In a software product development project at Nokia, there was a process called Nightly builds. Nightly builds can be thought of as a predecessor to Continuous Integration. It means that every night an automated system pulls the code added to the shared repository throughout the day and builds that code. The idea is quite similar to Continuous Integration, but since the code that was built at night was quite large, locating and fixing of bugs was a real pain. Due to this, Nokia adopted Continuous Integration (CI). As a result, every commit made to the source code in the repository was built. If the build result shows that there is a bug in the code, then the developers only need to check that particular commit. This significantly reduced the time required to release new software.

What is Continuous Integration?

Continuous integration is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error. In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If the deployment is a success, the code is pushed to Production. This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

What is Jenkins?

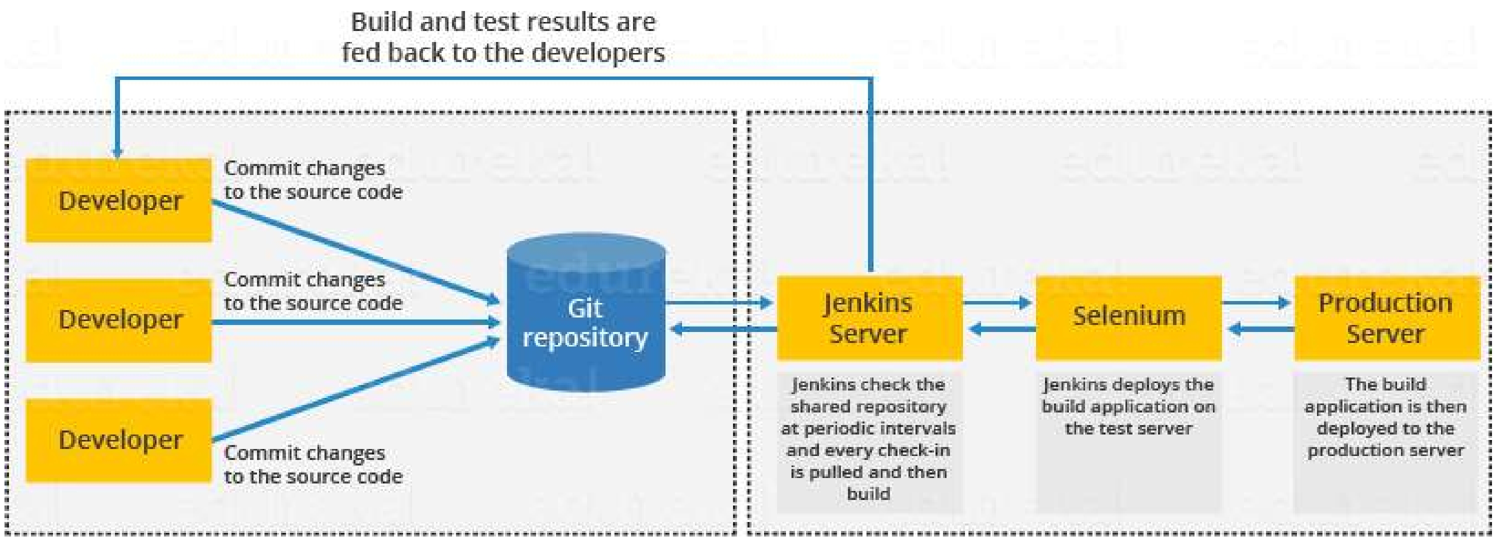
Jenkins is an open source automation tool written in Java programming language that allows continuous integration.

Jenkins builds and tests our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows us to continuously deliver our software by integrating with a large number of testing and deployment technologies.

Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins adds development life-cycle processes of all kinds, including build, document, test, package, stage, deploy static analysis and much more.

Let's see how Jenkins works


Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

Let's see how Jenkins works

First of all, a developer commits the code to the source code repository. Meanwhile, the Jenkins checks the repository at regular intervals for changes.

Soon after a commit occurs, the Jenkins server finds the changes that have occurred in the source code repository. Jenkins will draw those changes and will start preparing a new build.

If the build fails, then the concerned team will be notified.

If built is successful, then Jenkins server deploys the built in the test server.

After testing, Jenkins server generates a feedback and then notifies the developers about the build and test results.

It will continue to verify the source code repository for changes made in the source code and the whole process keeps on repeating.