https://www.youtube.com/playlist?list=PLhW3qG5bs-L_ZCOA4zNPSoGbnVQ-rp_dG

https://www.youtube.com/watch?v=p7-U1_E_j3w

Integration in software engineering means combining **software** parts (so-called subsystems) into one system.

Case 1:

For example, let say our organization bought HR management system for HR operation and that is built on ServiceNow. For employee attendance, we have biometric attendance system which developed in Java. The HR system's payroll module depends on employee attendance. So, if we want the complete system to be automated, then the attendance system must talk to the HR system. That's why integration is required.

Case 2: When acquiring software, many companies decide to buy only the components that they actually need in that moment. This way may be cheaper and thus seem more profitable at the beginning, but can very quickly become counterproductive. As your organisation evolves, you start using more and more independent tools, this results in productivity drop and inaccurate data analysis. Luckily, systems integration is here to save your business.

Systems integration is a great solution for companies who struggle with working on multiple independent subsystems and experiencing a lot of time being wasted due to the necessity of re-entering data to each of the tools manually.

Apart form Jenkins other powerfull tools ,Apache Gump, Buildbot, Bamboo, Travis CI

What is Continuous Integration?

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 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.

A Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to **host**, **monitor**, **compile** and **test** code, or code changes, like:

- **Continuous Integration Server** (Jenkins, Bamboo, CruiseControl, TeamCity, and others)
- **Source Control Tool** (e.g., CVS, SVN, GIT, Mercurial, Perforce, ClearCase and others)
- **Build tool** (Make, ANT, Maven, Ivy, Gradle, and others)
- **Automation testing framework** (Selenium, Appium, TestComplete, JUNIT,UFT, and others)

BankApplication :Server:Jenkins

SourceCode:Git:AccountDetails.LoginDetails-otp.NetbankingDetails

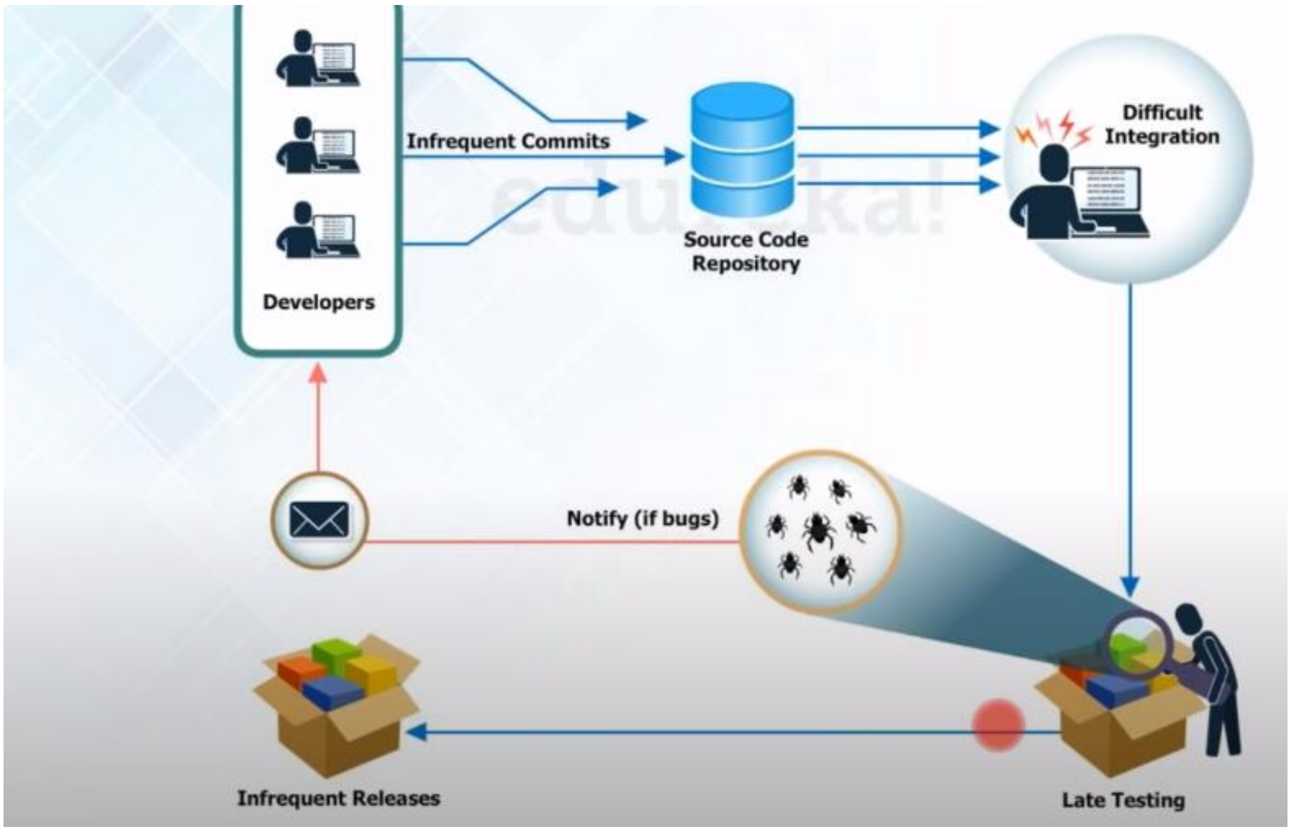
MavenProject :TestCases

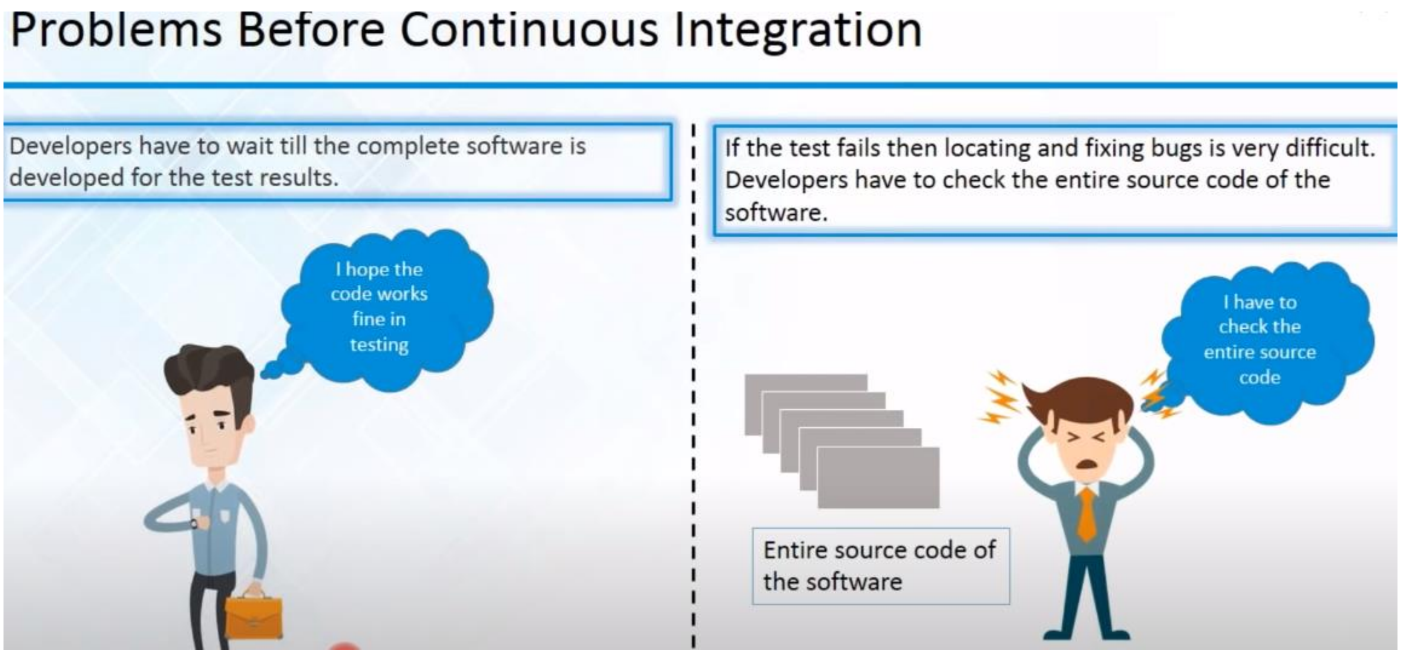
Jenkin History

- Kohsuke Kawaguchi, a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively. In 2004, created an automation server called Hudson that automates build and test task.
- In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open source community, so they forked Hudson and renamed it as Jenkins.
- Both Hudson and Jenkins continued to operate independently. But in short span of time, Jenkins acquired a lot of projects and contributors while Hudson remained with only 32 projects. With

time, Jenkins became more popular, and Hudson is not maintained anymore.

Jenkin : Is a Java Application – platform independent




I hope the code works fine in testing
I have to check the entire source code
Entire source code of the software

Problems Before Continuous Integration

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

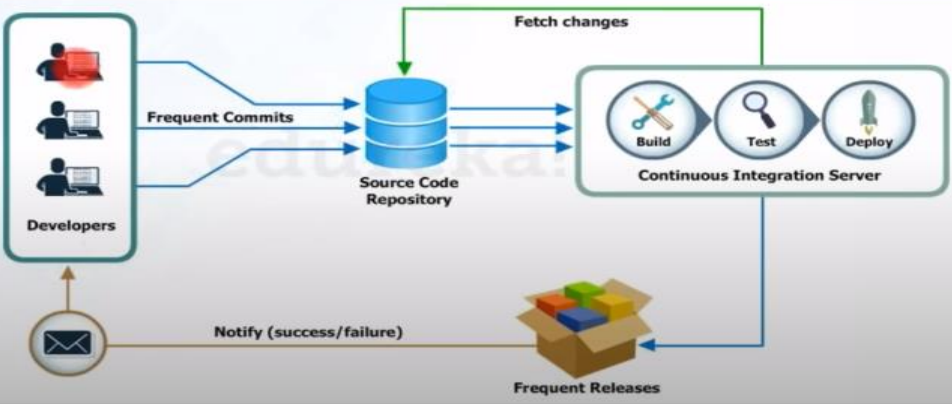
If the test fails then locating and fixing bugs is very difficult. Developers have to check the entire source code of the software.

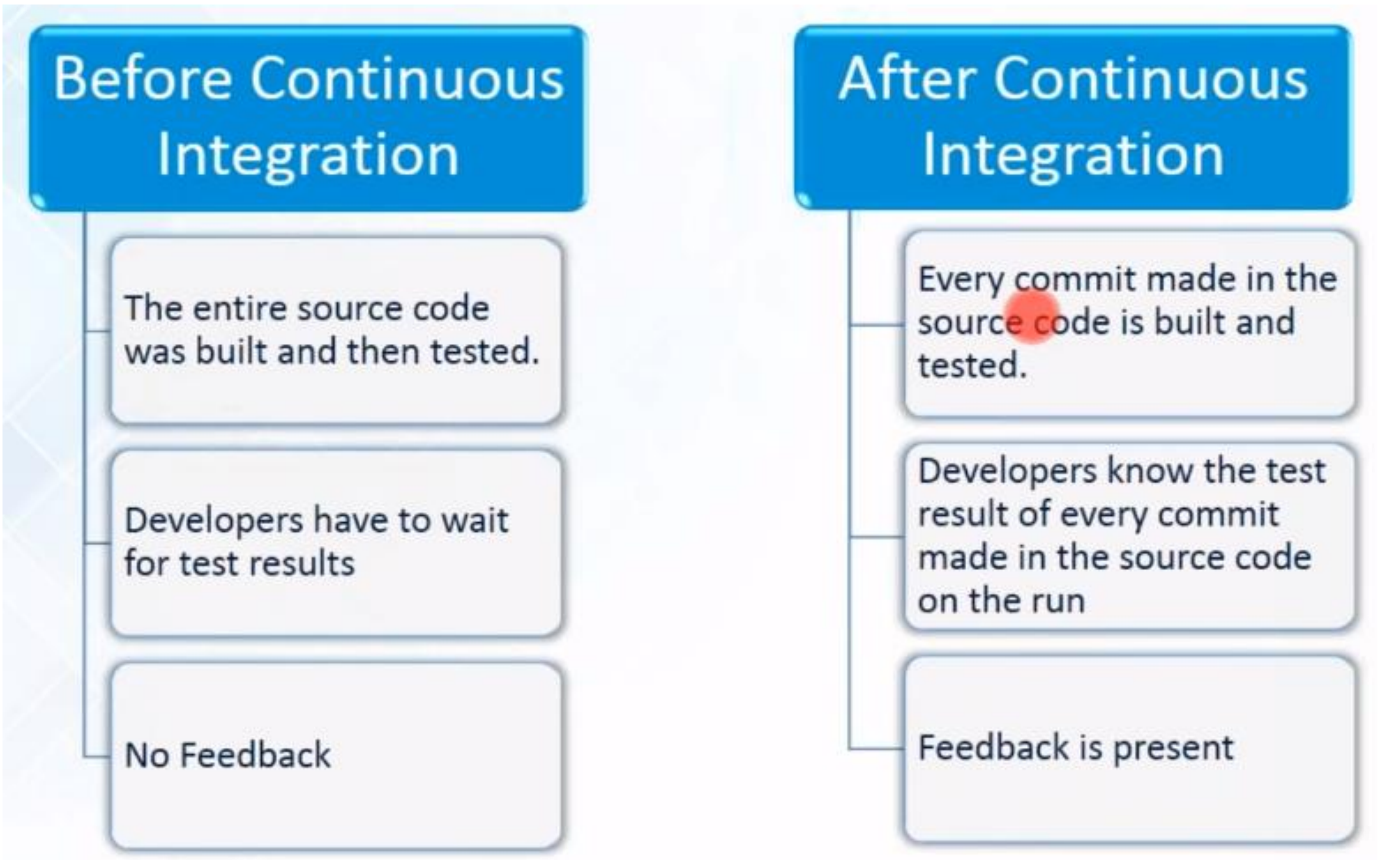
Software delivery process was slow

Continu
archite
not pre

Continuous Integration To The Rescue

- ❑ Since after every commit to the source code an auto build is triggered and then it is automatically depl
  server
- ❑ If the test results shows that there is a bug in the code then the developers only have to check the last
  the source code
- ❑ This also increases the frequency of new software releases
- ❑ The concerned teams are always provided with the relevant feedback



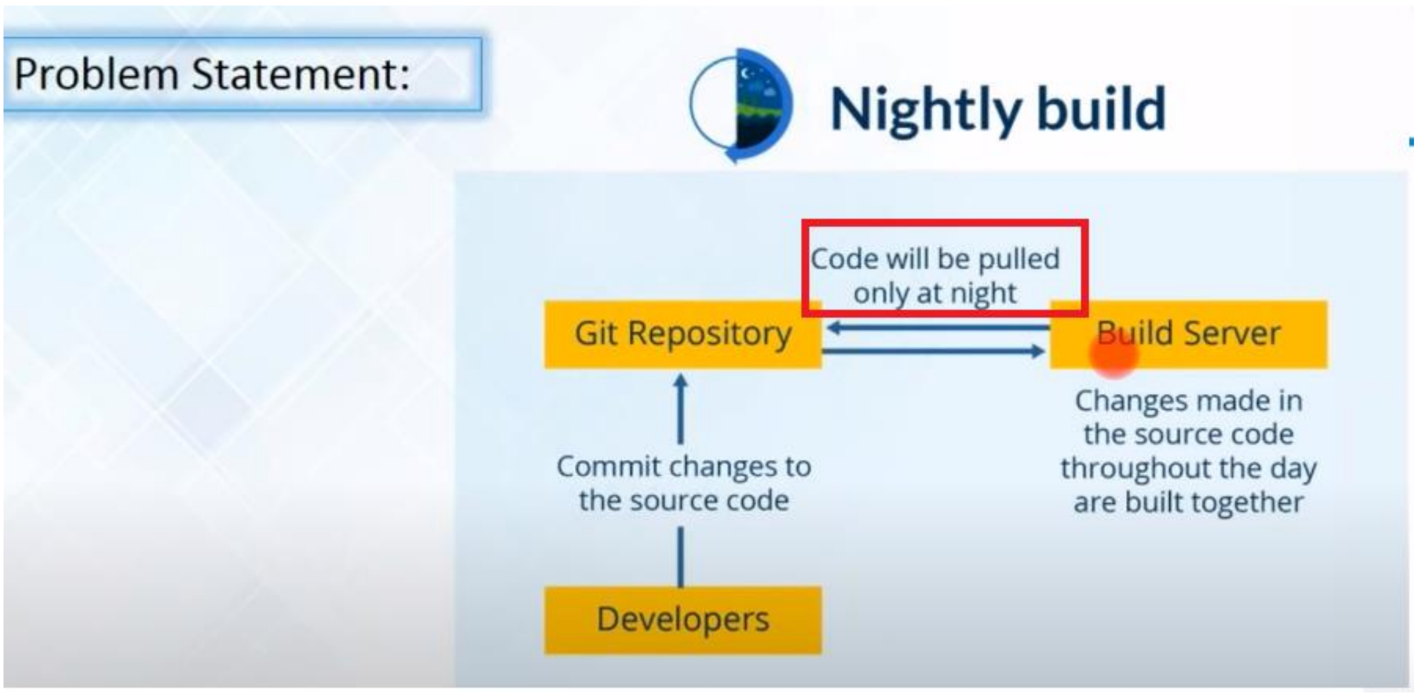


Before Continuous Integration

- The entire source code was built and then tested.
- Developers have to wait for test results
- No Feedback

After Continuous Integration

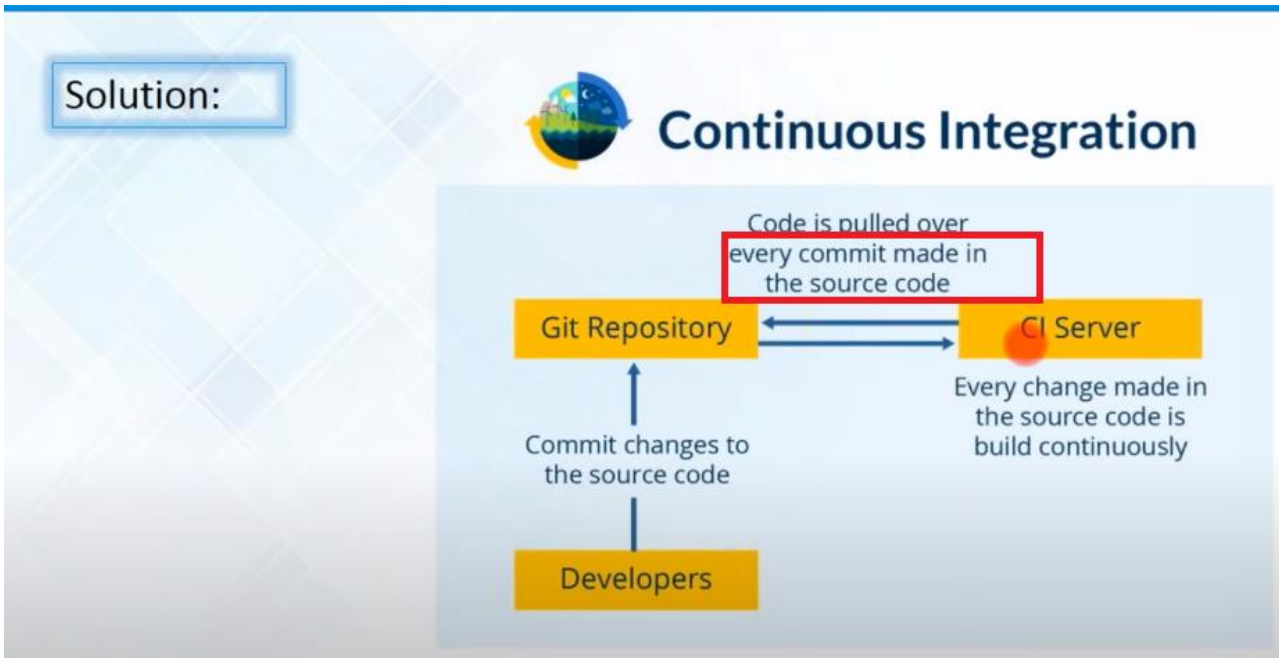
- Every commit made in the source code is built and tested.
- Developers know the test result of every commit made in the source code on the run
- Feedback is present



Problem Statement:

Nightly build

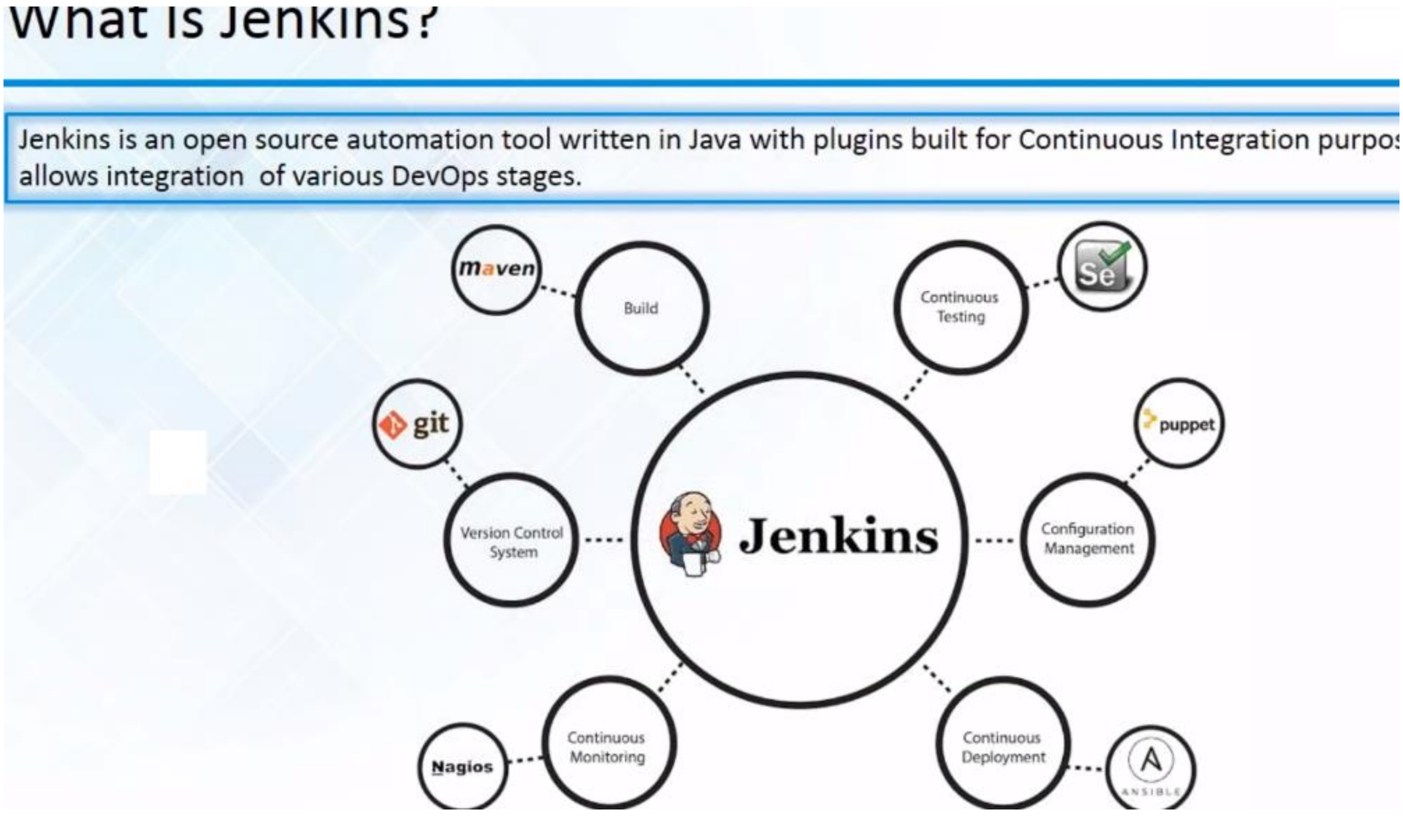
Code will be pulled only at night
Git Repository
Build Server
Commit changes to the source code
Developers
Changes made in the source code throughout the day are built together



Solution:

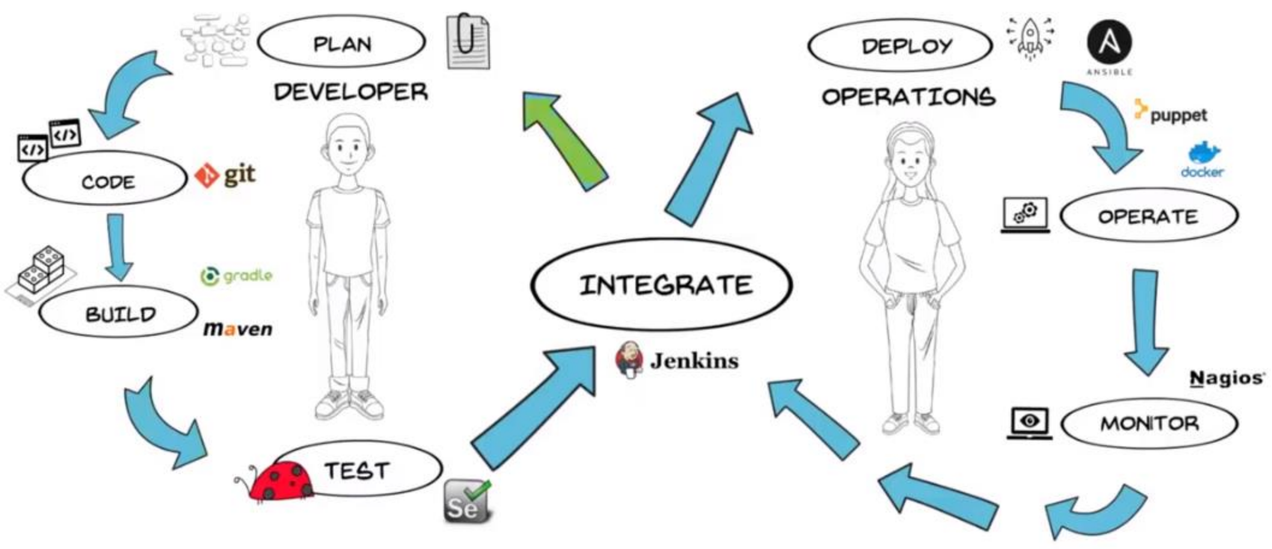
Continuous Integration

Code is pulled over every commit made in the source code
Git Repository
CI Server
Every change made in the source code is build continuously
Commit changes to the source code
Developers



What is Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpos
allows integration of various DevOps stages.



Before Jenkins	After Jenkins
Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed. Code commit built, and test cycle was very infrequent, and a single build was done after many days.	The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team. If the build fails, then Jenkins will notify the errors to the developer team.
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build	The code is built immediately after any of the Developer commits.

It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it's easy to detect whose code caused the built to fail
Code build and test process are entirely manual, so there are a lot of chances for failure.	Automated build and test process saving timing and reducing defects.
The code is deployed once all the errors are fixed and tested.	The code is deployed after every successful build and test.
Development Cycle is slow	The development cycle is fast. New features are more readily available to users. Increases profits.

Disadvantages of using Jenkins

Though Jenkins is a very powerful tool, it has its flaws.

- Its interface is out dated and not user friendly compared to current UI trends.
- Though Jenkins is loved by many developers, it's not that easy to maintain it because Jenkins runs on a server and requires some skills as server administrator to monitor its activity.
- One of the reasons why many people don't implement Jenkins is due to its difficulty in installing and configuring Jenkins.
- **Continuous integrations regularly break due to some small setting changes. Continuous integration will be paused and therefore requires some developer attention.**