



# **JENKINS**

**Jenkins for Continuous Integration**

- Jenkins is the most popular, open source Continuous Integration tool. It has tons of plugins that enhance its functionality.
- As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.



# WHAT IS JENKIN FOR CONTINUOUS INTEGRATION?

- What is Continuous Integration?
- What is Jenkin?
- Jenkin History
- Why Continuous Integration with Jenkins?
- Real-world case study of Continuous Integration
- Advantages of using Jenkins
- Disadvantages of using Jenkins



# 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, UFT, and others)



# CI Benefits

- Immediate bug detection
- No integration step in the life-cycle.
- A deployable system at any given point.
- Record of evolution of the project.
- Detect system development problems earlier.
- Reduce risks of costs, schedule, and budget.
- Measurable and visible code quality.



# WHAT IS JENKIN?

- Jenkins is an open source automation server written in Java.
- Detects changes in Subversion/GIT.., performs tasks, repeatedly.(Build, Test, Deploy, Package, Integrate)
- A fork of the original Hudson Project.
- Plug-in extensibility.
- Under development since 2005.
- <http://jenkins-ci.org/> → <https://jenkins.io/>



- It is a server-based application and requires a web server like Apache Tomcat.
- The reason Jenkins became so popular is that of its monitoring of repeated tasks which arise during the development of a project.
- For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.
- By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate.
- Jenkins supports the complete development lifecycle of software from building, testing, documenting the software, deploying and other stages of a software development lifecycle.





# 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 the 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.



# WHY CONTINUOUS INTEGRATION WITH JENKINS?

Before Jenkins	After Jenkins
<p>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.</p>	<p>The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day</p>
<p>Code commit built, and test cycle was very infrequent, and a single build was done after many days.</p>	<p>If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team.</p> <p>If the build fails, then Jenkins will notify the errors to the developer team.</p>



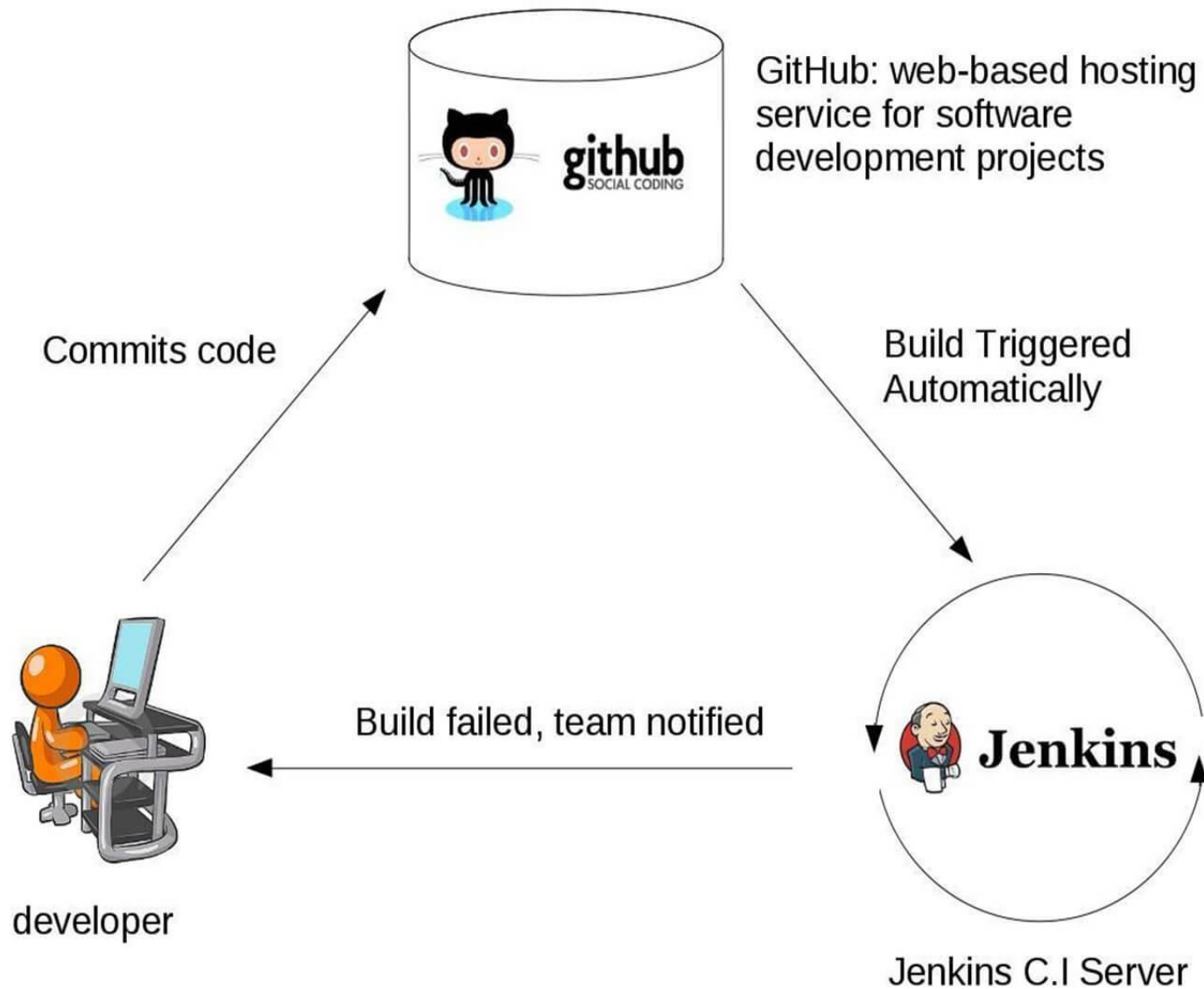
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build	The code is built immediately after any of the Developer commits.
It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it's easy to detect whose code caused the build 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.



# REAL-WORLD CASE STUDY OF CONTINUOUS INTEGRATION

- Old phone Nokia used to implement a procedure called **nightly build**. After multiple commits from diverse developers during the day, the software built every night. Since the software was built only once in a day, it's a huge pain to isolate, identify, and fix the errors in a large code base.
- Later, they adopted **Continuous Integration approach**. The software was built and tested as soon as a developer committed code. If any error is detected, the respective developer can quickly fix the defect.





# JENKINS PLUGINS

- With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery tool chain.
- By default, Jenkins comes with a limited set of features.
- If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git.
- In fact, for integration with tools like Maven, Amazon EC2, you need to install respective plugins in your Jenkins.



[Jenkins](#)

 [New Job](#)

 [Manage Jenkins](#)

 [People](#)

 [Build History](#)

## Build Queue

No builds in the queue.

## Build Executor Status

#	Status
1	Idle
2	Idle

## Manage Jenkins



[Configure System](#)

Configure global settings and paths.



[Reload Configuration from Disk](#)

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files di



[Manage Plugins](#)

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



[System Information](#)

Displays various environmental information to assist trouble-shooting.



[System Log](#)

System log captures output from `java.util.logging` output related to Jenkins.



[Load Statistics](#)

Check your resource utilization and see if you need more computers for your builds.



[Jenkins CLI](#)

Access/manage Jenkins from your shell, or from your script.



[Script Console](#)

Executes arbitrary script for administration/trouble-shooting/diagnostics.



[Manage Nodes](#)

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



[Install as Windows Service](#)

Installs Jenkins as a Windows service to this system, so that Jenkins starts automatically when the machine boots.



[Prepare for Shutdown](#)

Stops executing new builds, so that the system can be eventually shut down safely.



# ADVANTAGES OF USING JENKINS

- Jenkins is being managed by the community which is very open. Every month, they hold public meetings and take inputs from the public for the development of Jenkins project.
- As technology grows, so does Jenkins. With plugins, Jenkins becomes even more powerful and feature rich.
- Jenkins also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.
- The reason why Jenkins became popular is that it was created by a developer for developers.





# 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.



# CONCLUSION

- In Continuous Integration, after a code commit, the software is built and tested immediately
- Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions
- Before Jenkins when all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.



- After Jenkins the code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day
- By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git
- The biggest pros of Jenkins is that it is managed by the community which holds public meetings and take inputs from the public for the development of Jenkins projects
- The biggest con of Jenkin is that Its interface is out dated and not user friendly compared to current UI trends.

