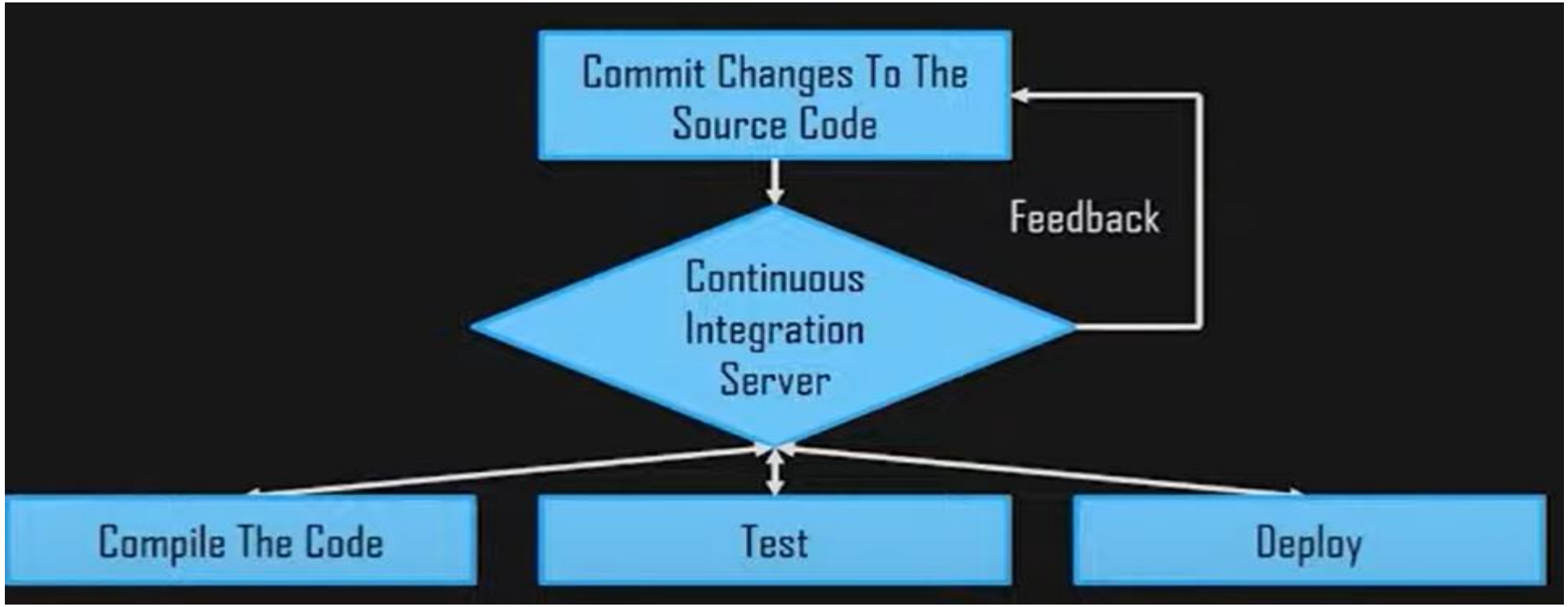
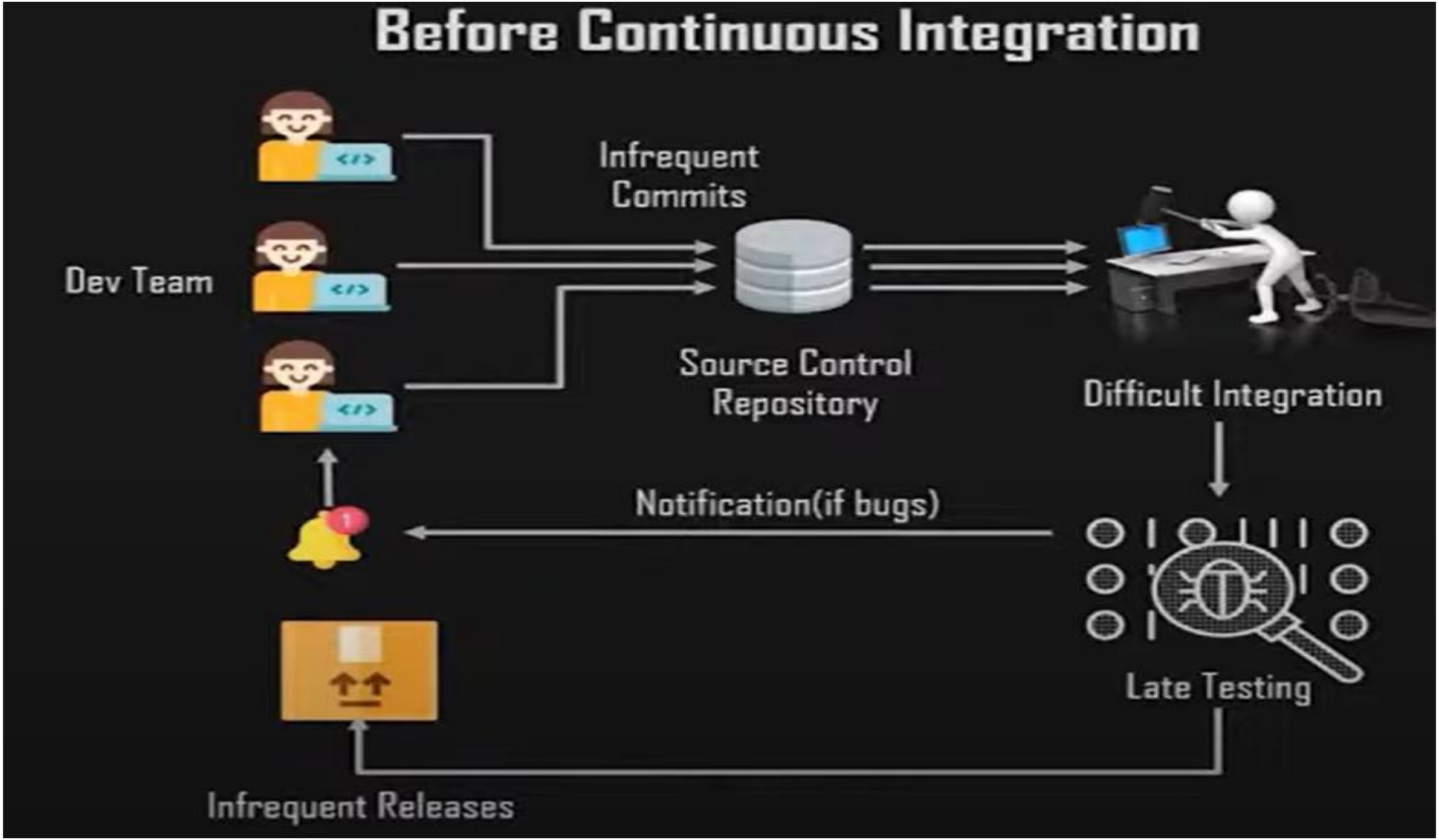
Welcome to Jenkins

What is Continuous Integration

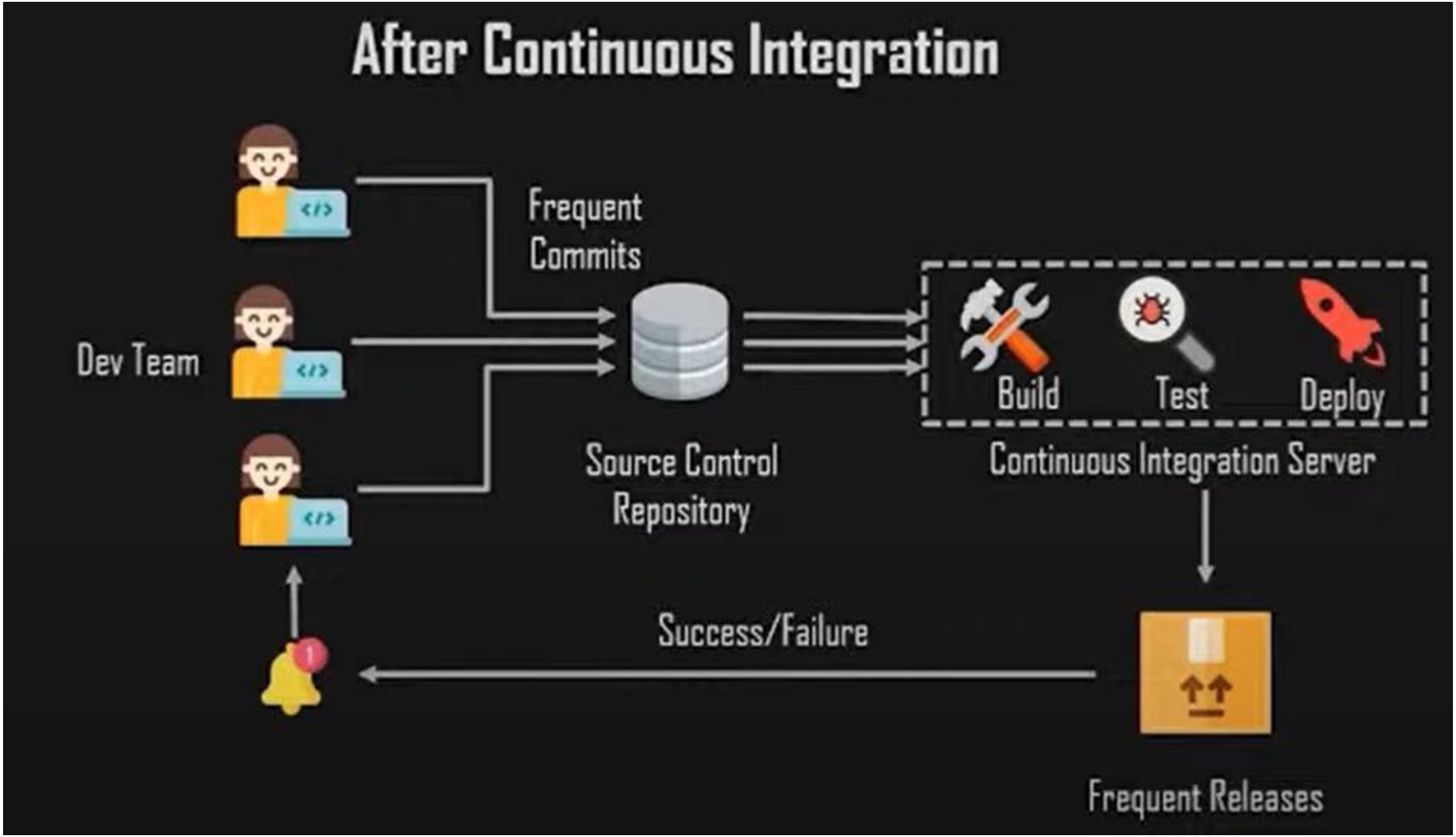
- Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.
- Every commit made in the repository is then built. This allows the teams to detect the problems early. Apart from this, depending on the Continuous Integration tool, there are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.




Before Continuous Integration
Infrequent
Commits
Dev Team
Source Control
Repository
Difficult Integration
Notification(if bugs)
Late Testing
Infrequent Releases

Issues faced before Continuous Integration

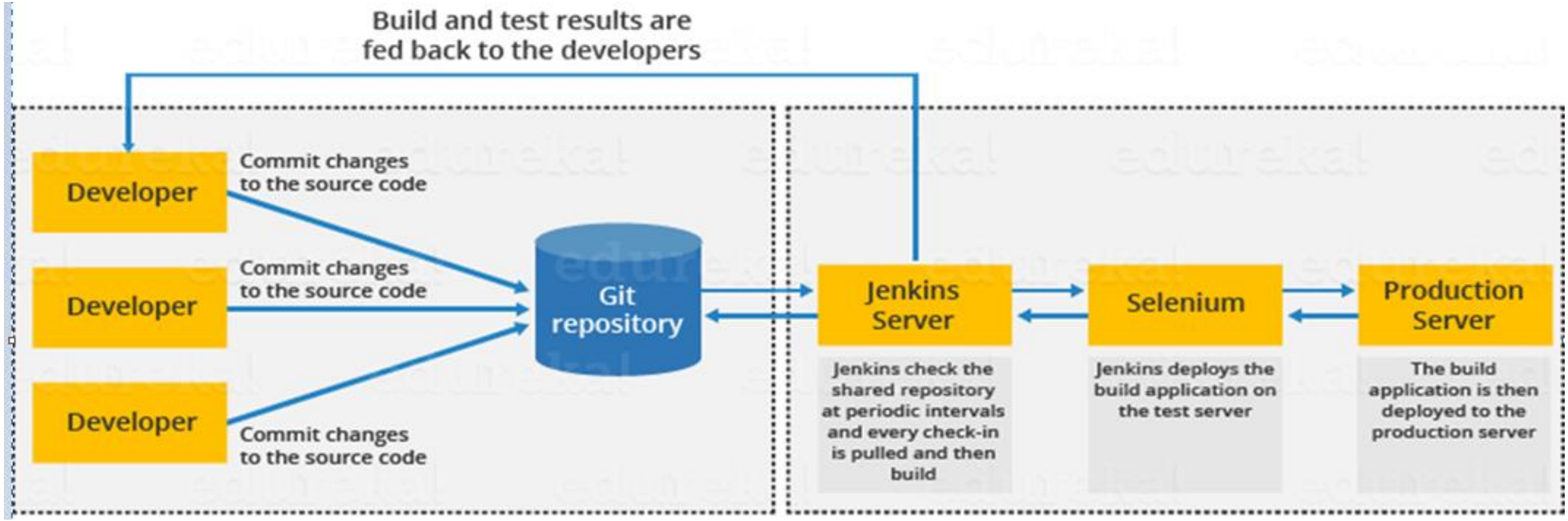
- Long Waits to test code
- Difficult to Debug
- Slow Software Delivery process
- Infrequent feedback cycles


After Continuous Integration
Frequent
Commits
Dev Team
Source Control
Repository
Build
Test
Deploy
Continuous Integration Server
Success/Failure
Frequent Releases

Introduction to Jenkins

- Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
- Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose.
- Jenkins is used to build and test your software projects 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 you to continuously deliver your software by integrating with a large number of testing and deployment technologies.
- Continuous Integration is the most important part of DevOps that is used to integrate various DevOps stages.
- Jenkins is the most famous Continuous Integration tool.
- Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven etc.

Jenkins Architecture



History of Jenkins

- The Jenkins project was started in 2004 (originally called Hudson) by Kohsuke Kawaguchi, while he worked for Sun Microsystems.
- Kohsuke was a developer at Sun and got tired of incurring the wrath of his team every time his code broke the build.
- The Hudson was renamed to Jenkins in 2011 after a dispute with Oracle, which had forked the project and claimed rights to the project name.
- The Oracle fork, Hudson, continued to be developed for a time before being donated to the Eclipse Foundation. Oracle's Hudson is no longer maintained and was announced as obsolete in February 2017.
- On April 20, 2016 version 2 was released with the Pipeline plugin enabled by default.
- The plugin allows for writing build instructions using a domain specific language based on Apache Groovy.

Other CI Tools

- Jenkins 52%
- TeamCity 6.83%
- Bamboo 1.76%
- Buddy
- GitLab CI
- CircleCI 5.61%
- TravisCI 1.51%

Jenkins Setup

- Follow the Instructions given in below URL

Jenkins download and deployment

- Once Jenkins is installed, it runs on port no 8080 by default

  - Access the Jenkins from browser
    - http://<IP Address or localhost>:8080
  - Unlock Jenkins
  - Install Plugins
  - Create Admin user

Plugins in Jenkins

- As of cut-off in 2021, the Jenkins plugin repository had over 2,000 plugins available for users to download and install.
- Jenkins → Manage Jenkins → Manage Plugins
- Plugins are with extension of
  - hpi or
  - jpi

Jenkins Job types

- Jenkins is a flexible and highly configurable automation server that can be used to automate many different types of tasks. Some common types of jobs in Jenkins include:
- **Freestyle project**: a basic type of Jenkins job that provides a wide range of options for customizing and configuring the build process.
- **Pipeline:** a type of Jenkins job that uses a Groovy script to define the entire build process, from building and testing the code to deploying it to production.
- **Multi-configuration project:** a type of Jenkins job that allows for running the same build with different configurations.
- **Folder:** a type of Jenkins job that allows you to organize and manage multiple jobs in a hierarchical structure.
- **Maven project:** a type of Jenkins job that is optimized for building and testing Maven-based projects.
- **Multi-Branch project:** a type of Jenkins job that allows for running the same build with different branches.

CRON Jobs in Jenkins

- Cron is the baked in task scheduler - run things at fixed times, repeat them etc. In fact, Jenkins uses something like the cron syntax when you are specifying specific times you want a Job to run.
- Syntactically, CRON contains 5 places for each time entry, thus a conventional one-liner syntax of the CRON template would be somehow like:

0 23 * * *
// Staged as
{Minute} {Hour} {DayOfMonth} {Month} {DayofWeek}

CRON Jobs in Jenkins

- **Minute (0-59)** – tells at what minute of the hour the Job should build.
- **Hour (0-23 With 0 is the midnight)** - tells at what hour the job should build in the 24-hour clock.
- **Day of the month (1-31)** - tells on which day of the month your job should build, e.g., 23rd of each month.
- **Month (1-12)** - tells the month when your job should build. You can number or name the month as well. i.e., April, May, etc.
- **Day of the week (0-7 With 0 or 7 both being Sunday)** - tells the day of the week when you want to build your job. It can also be a number or name like Mon etc.

- // Every day at 6 in the evening
- 0 18 * * *
- You need to make the first entry (minute) fixed if you want to work on the hour. The above expression means to run your job at the 0th minute of 18th hour DAILY without any specification of day, month or week.

CRON Jobs in Jenkins

- After every 20 minutes
  *0/20 * * * *
- After every 4 hours
  0 */4 * * *
  // OR
  0 0,4,8,12,16,20 * * *
- In the first case, it will run at 0th minute, 20 minutes after, 40 minutes after means after every 20 minutes. 20 is the offset here which tells after how many minutes this should trigger.
- Similarly, in the next example, the first entry (minute) is fixed and the second entry (hour) contains an offset which tells keep repeating it after 4 hours. In the next line, the less verbose and more readable syntax of the same CRON is also given.

- Every day at 11 in the morning and at night both
  0 11, 22 * * *
- You can define the multiple values for same entry using a comma. Here, the hour place has two values, first for 11 am and the next for 11 pm.

CRON Jobs in Jenkins

- Every Saturday and Sunday at 5 PM

  0 17 * * 6,7
- This will run every Saturday (6) and Sunday (7) at 5 pm (17).

Jenkins Executors

- In Jenkins, an executor is a process that performs a build on a node (also called a worker or slave). An executor runs a build and executes the various build steps defined in a Jenkins job.
- Jenkins is designed to run multiple builds simultaneously, so it uses multiple executors to run builds in parallel. This allows Jenkins to make efficient use of resources and finish builds more quickly.
- The number of executors that a node has can be configured in the Jenkins system configuration. By default, a node will have two executors, but you can configure additional executors as needed. The total number of executors that Jenkins has available is determined by the total number of executors across all nodes in the system.
- It's worth noting that while having more executors can allow Jenkins to run builds more quickly, it can also put additional strain on your system resources, such as memory and CPU. As a result, it's important to carefully consider the number of executors that you have available, and adjust as needed based on the needs of your system.

Email Configuration in Jenkins

- Manage Jenkins → Configure System
- At the bottom of the page you can see Email Notification
- Provide below details
  - SMTP Server Name
  - Default email suffix
  - Username
  - Password
  - Port
- Once Email Configuration is successful then Go the Job → Configure
- Post Build action → Email Notification and provide recipient email details separated by space

Q & A