Agile Software Development (TCS 855)

Unit-III Agile Software Design and PROGRAMMING

Continuous Integration, Automated Build tools and version control

Prof.(Dr.) Santosh Kumar

Department of Computer Science and Engineering

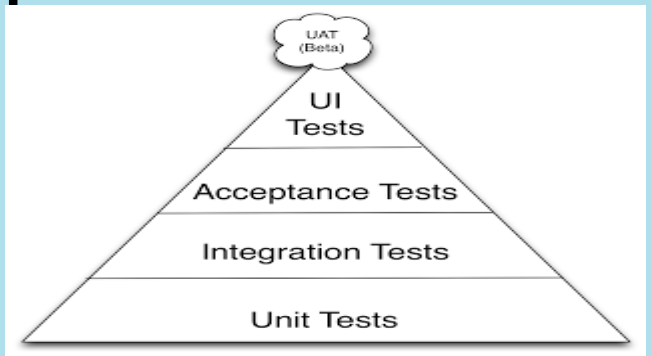
Graphic Era Deemed to be University, Dehradun

Continuous Integration

- Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project.
- It’s a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run.
- Automated tools are used to assert the new code’s correctness before integration.
- A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

- The first step on your journey to continuous integration: setting up automated testing.
- **Automated testing:** To get the full benefits of CI, you will need to automate your tests to be able to run them for every change that is made to the main repository.
- There are many types of tests implemented, but it is not necessary to do everything at once if you're just getting started. You can start small with unit tests and work on extending your coverage over time
  - ✓ **Unit tests** are narrow in scope and typically verify the behaviour of individual methods or functions.
  - ✓ **Integration tests** make sure that multiple components behave correctly together. This can involve several classes as well as testing the integration with other services.
  - ✓ **Acceptance tests** are similar to the integration tests but they focus on the business cases rather than the components themselves.
  - ✓ **UI tests** will make sure that the application functions correctly from a user perspective.

- Not all the tests are equal, and you can visualize the tradeoffs that you will make with the test pyramid developed by **Mike Cohn**



- **Continuous integration in 5 steps**
- a good idea of the concepts behind continuous integration, and we can boil it down to this:

1. Start writing tests for the critical parts of your codebase.
2. Get a CI service to run those tests automatically on every push to the main repository.
3. Make sure that your team integrates their changes everyday.
4. Fix the build as soon as it’s broken.
5. Write tests for every new story that you implement.

Automated Build tools

- Automated Build Tool is a software that compiles the source code to machine code.
- Automation tools are used to automate the whole process of software build creation and the other related processes like packaging binary code and running the automated tests.
- These automation tools can be categorized into two types i.e. Build-Automation Utility and Build-Automation servers.
- Build automation utilities perform the task of generating build artifacts. **Maven and Gradle** come under this category of build automation tools.
- There are three types of Build Automation servers i.e. **On-demand automation, Scheduled automation, and Triggered automation.**

- List of the Top Build Automation ToolsComparison of the Best Automated Build Deployment Software

#1) Jenkins

#2) Maven

#3) Gradle

#4) Travis CI

#5) Bamboo

#6) CircleCI

#7) TeamCity

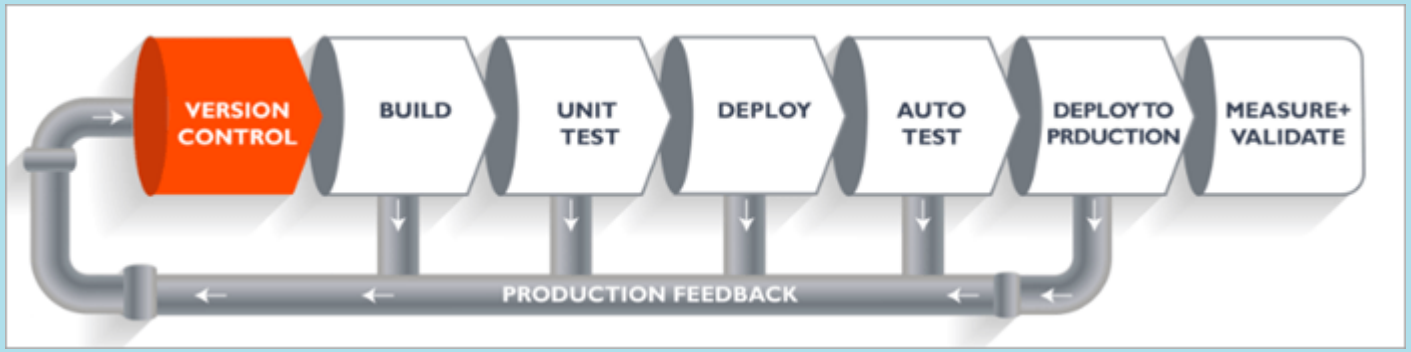
#8) Apache Ant

#9) BuildMaster

#10) Codeship

Build Deployment And Continuous Integration Process

- to implement Continuous Integration and Continuous Deployment then adopting the Build tool will be the first step of it.
- Build Tools provide the features of an extensive library of plugins, build & source code management functionalities, dependency management, parallel testing & build execution, and compatibility with IDE.
- **The complete process of Build Automation, Continuous Integration and Continuous Deployment is shown in the below image.**



Challenges for Build Automation

#1) Longer builds: Longer builds take more time to run, it will increase the developer's wait time and thereby reduces productivity.

#2) Large volumes of builds: If a large volume of builds is running, then you will get limited access to the build servers for that specific period.

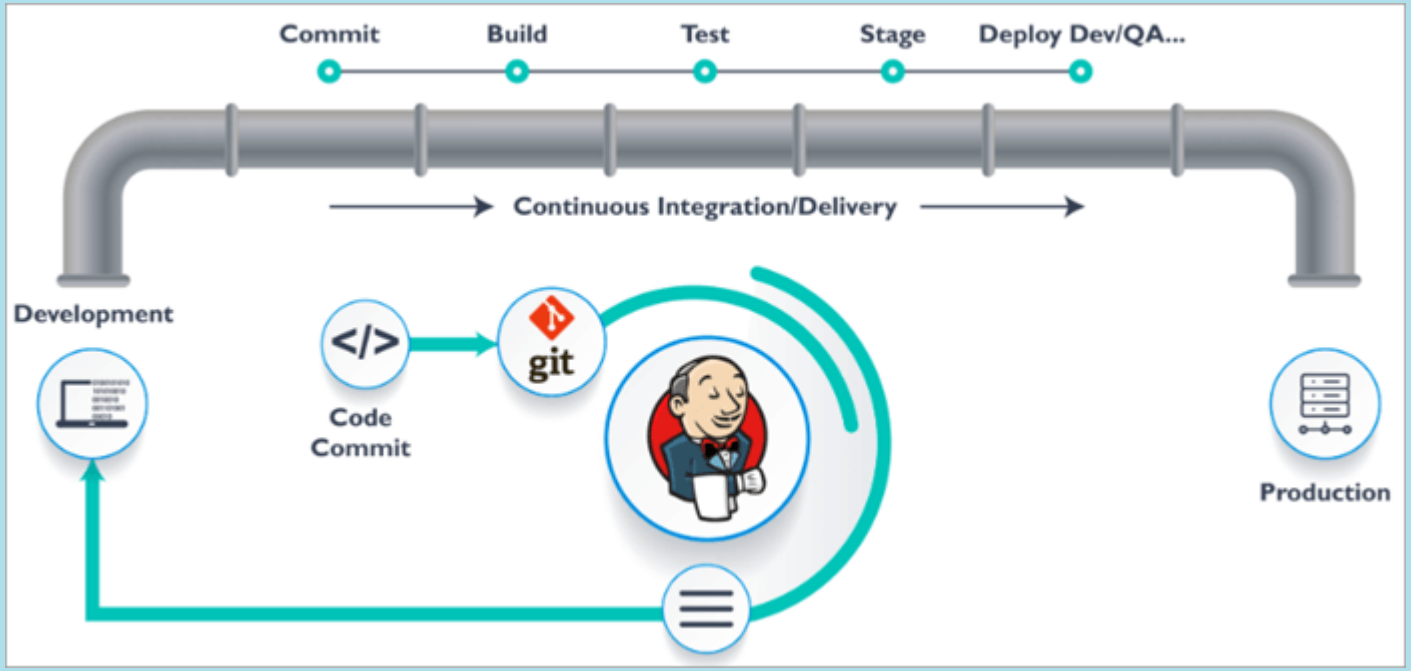
#3) Complex builds: Complex builds may require extensive manual efforts and may reduce flexibility.

Benefits Of Automation Build Tools

Using the build automation software has several benefits as mentioned below:

- Saving time and money.
- Keeping a history of builds and releases. It will help in investigating the issue.
- Dependencies on key personnel will be eliminated through these tools.
- It will accelerate the process.
- It will perform redundant tasks.

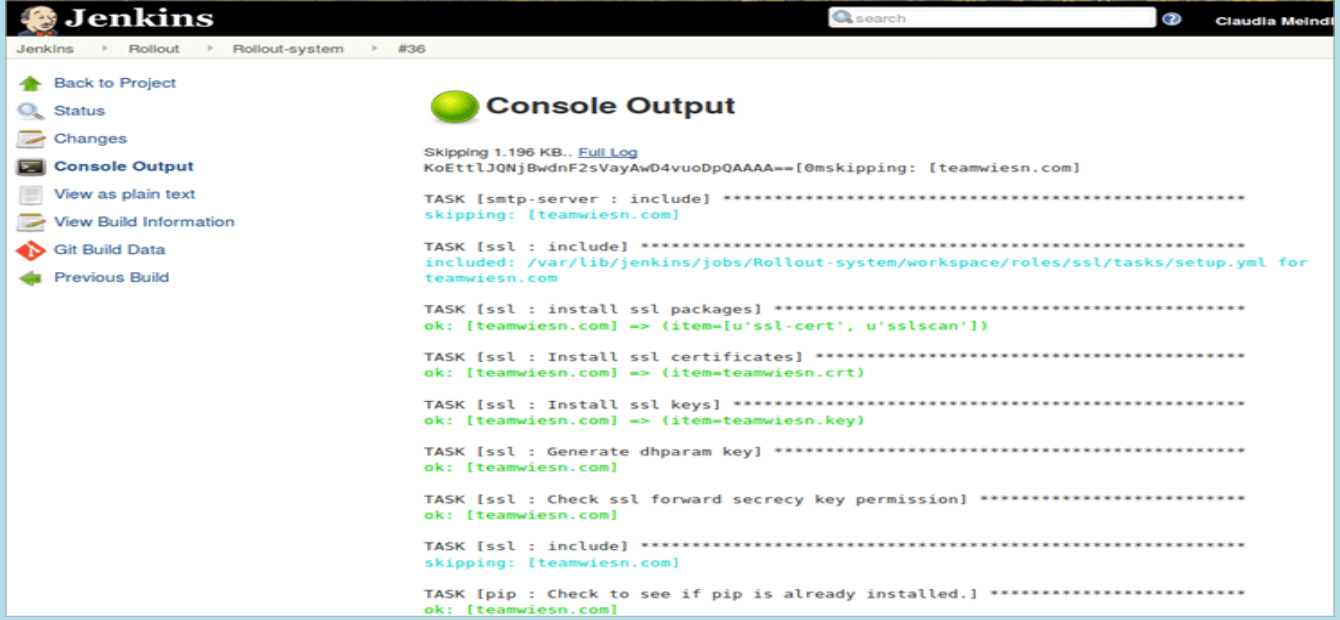
- **The complete automation of the software development process is explained in the below image. Here it is explained through the Jenkins tool as it is our top-rated build automation Software.**



#1) Jenkins

Best for small to large businesses.

Price: Free



Jenkins is an open-source tool. It can perform the task of building, testing, and deploying software. The platform is easy to install. For any project, Jenkins will work as a CI server and as a continuous delivery hub. It has features of extensibility and easy configuration.

Features:

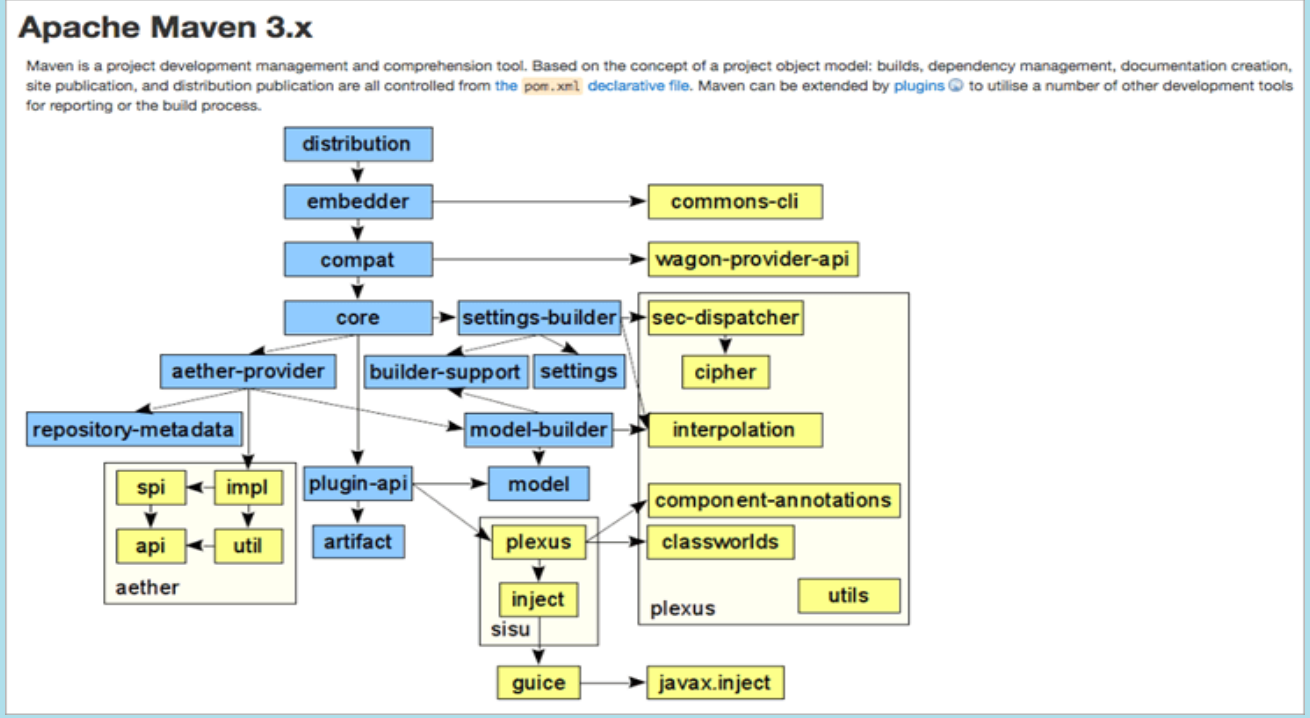
- Testing of isolated changes in a larger codebase.
- Automation of testing of builds.
- Work Distribution.
- Automation of software deployment.

Verdict: You will get good community support for Jenkins. It supports all major platforms. It can test and deploy on multiple platforms at a fast rate. It can distribute the work across multiple machines.

#2) Maven

Best for small to large businesses

Price: Free

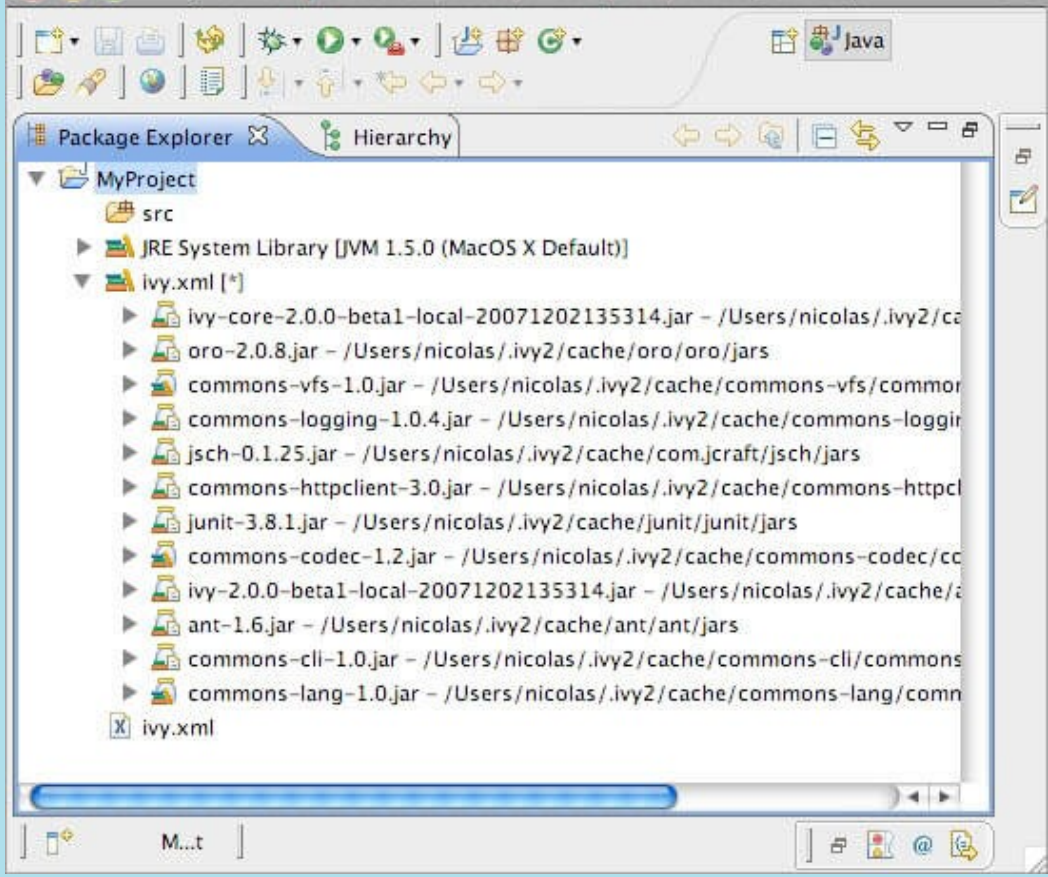


- Maven is an application that provides functionalities for project management. It has functionalities for project building, reporting, and documentation. You will be able to access the new features instantly. It is extensible through plugins. There will be no limitation on building the number of projects into a JAR, WAR, etc.
- **Features:**
- It supports working on multiple projects simultaneously.
- There will be consistent usage for all projects.
- It has features for dependency management.
- It provides a large and growing repository of libraries and metadata.
- It provides functionality for release management: It can distribute individual outputs.
- For managing the releases and distributing the publications, Maven will get integrated with your system. No additional configuration will be required for this.
- **Verdict:** As per the customer reviews, the tool is good for build automation and dependency management. For dependency management, it provides support to the central repository of JARs.

#3 Apache Ant

Best for individuals and businesses.

Price: Free



- Apache Ant is used to compile, assemble, test, and run Java applications. It has features for combining builds and dependency management. It will allow you to develop your antlibs. Antlibs will include Ant tasks and types.
- **Features:**
- It has various built-in tasks for compiling, assembling, testing, or running java application.
- No forcing of coding conventions.
- It provides a lot of ready-made commercial and open-source antlibs.
- It is a flexible platform.
- **Verdict:** Apache Ant is an open-source command-line tool. The tool is written in Java and gives its users the freedom to create their antlibs.

Version Control

- Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done to the code.
- **Why Version Control system is so Important?**
- A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what change has been made.
- A separate branch is created for every contributor who made the changes and the changes aren’t merged into the original source code unless all are analyzed as soon as the changes are green signalled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Benefits of the version control system:

a) Enhances the project development speed by providing efficient collaboration,

b) Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,

c) Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,

d) Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this **VCS,**

e) For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is **Git, Helix core, Microsoft TFS,**

f) Helps in recovery in case of any disaster or contingent situation,

g) Informs us about Who, What, When, Why changes have been made.

Use of Version Control System:

- **A repository:** It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
- **Copy of Work (sometimes called as checkout):** It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.

Types of Version Control Systems:

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems
- **Local Version Control Systems:** It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.
- **Centralized Version Control Systems:** Centralized version control systems contain just one repository and each user gets their own working copy. You need to commit to reflecting your changes in the repository. It is possible for others to see your changes by updating.
- Two things are required to make your changes visible to others which are:

- You commit
- They update