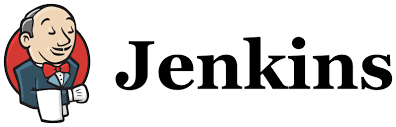
**Certified Jenkins Engineer**

****

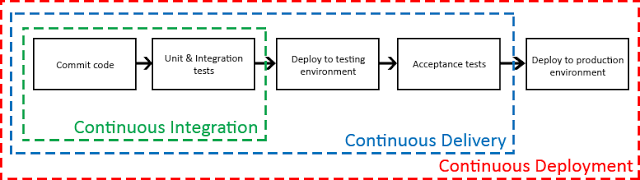
**CI- Continuous Integration**

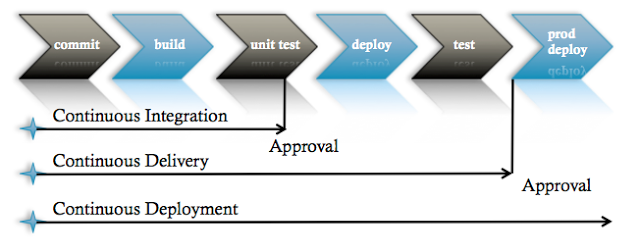
**CD- Continuous Delivery**

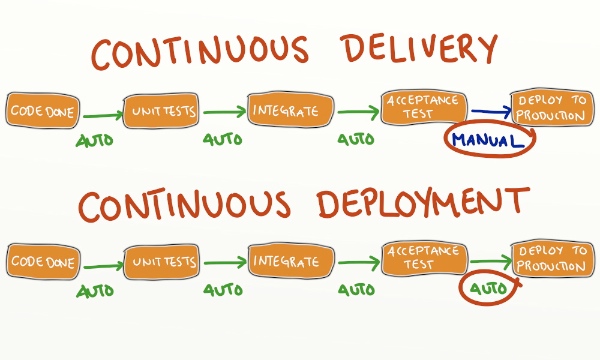
**CD-Continuous Deployment**

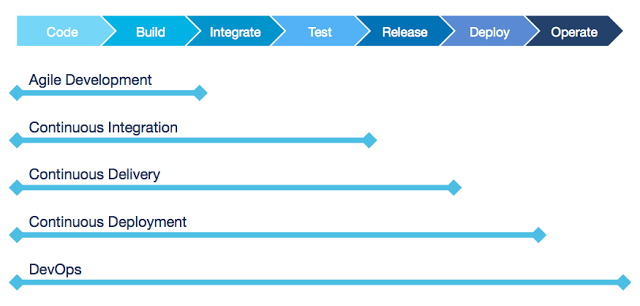
**Continuous(Integration,Deployment,Delivery)**

**Continuous Integration:(CI)**  
CI is a development practice that requires developers to integrate code into a shared repository(SVN/Clearcase/GIT) several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.  
 **Continuous Delivery:**  
Continuous Delivery is a **software development discipline where you build software in such a way that the software can be released to production at any time.** You're doing continuous delivery when: Your software is deployable throughout its life cycle.  
 **Continuous Deployment:**  
Continuous deployment is the next step past continuous delivery, where you are not just steadily creating a deployable package, but you are actually deploying it steadily  
  
Continuous Deployment is also **consistently deploying code to production** as features are completed, and as soon as you have met the release criteria for those features. That release criteria depends on your situation, and may be running some automated tests, code reviews, load tests, manual verification by a QA person or business stakeholder, or just having another pair of eyes look at your feature and make sure it doesn't explode. Again, the specific criteria can vary, but the key idea is to have a **steadily flowing pipeline pushing changes to production**, always moving the code forward, and keeping the pipeline as short as realistically possible.

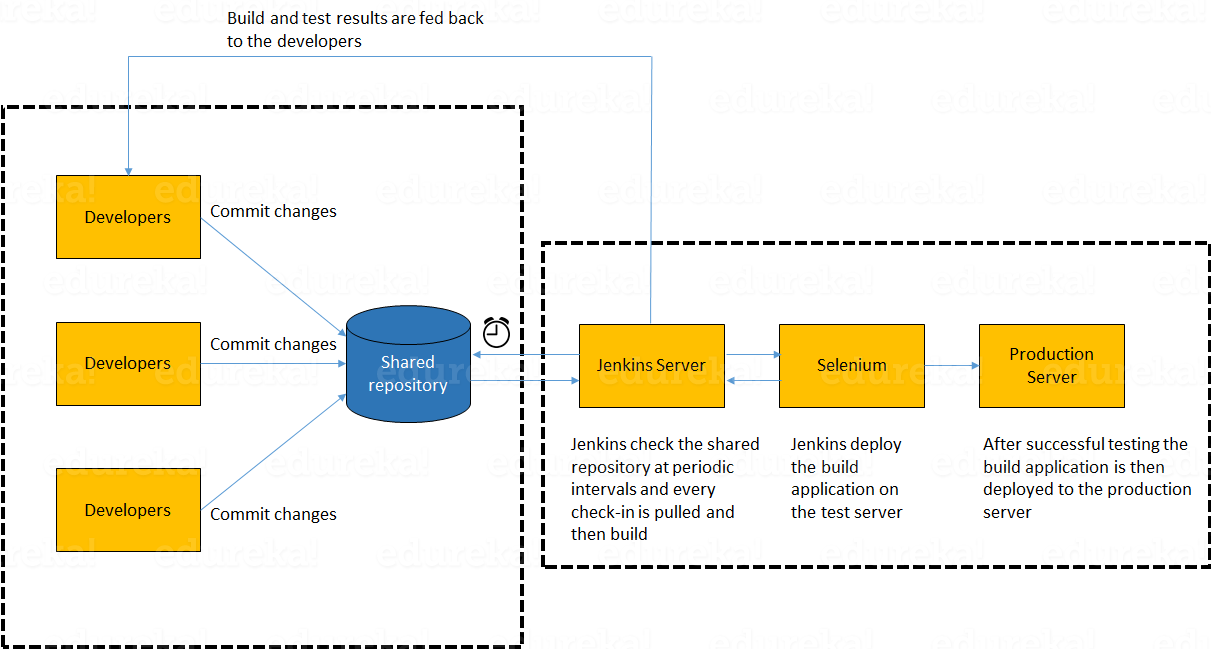








**What is meant by Continuous Integration**



In the diagram shown above:

1. Developers check out code into their private workspaces.
2. When they are done with it they commit the changes to the shared repository (Version Control Repository).
3. The CI server monitors the repository and checks out changes when they occur.
4. The CI server then pulls these changes and builds the system and also runs unit and integration tests.
5. The CI server will now inform the team of the successful build.
6. If the build or tests fails, the CI server will alert the team.
7. The team will try to fix the issue at the earliest opportunity.
8. This process keeps on repeating.