Introduction to CI/CD (Continuous Integration / Continuous delivery):

**Continuous Integration**: The practice of automatically building code periodically.

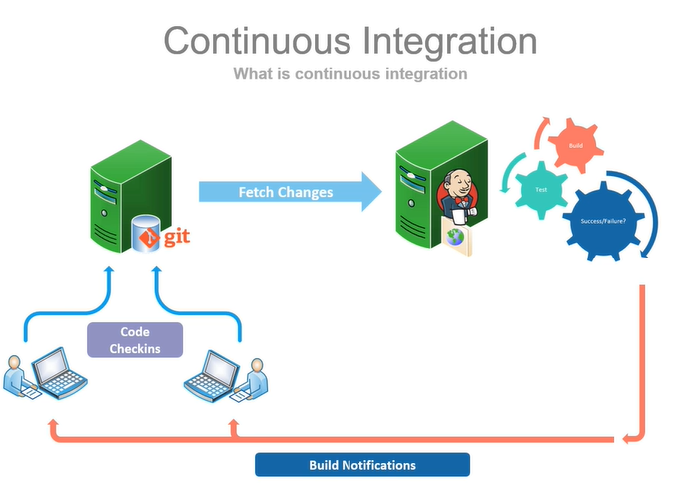
CI is the practice in software engineering of merging all developer working copies to a shared mainline several times a day.

The resulting binary after building the code can be deployed automatically to the target environment is called **continuous delivery or continuous deployments.**

CD is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing and releasing software faster.

**How CI works:**

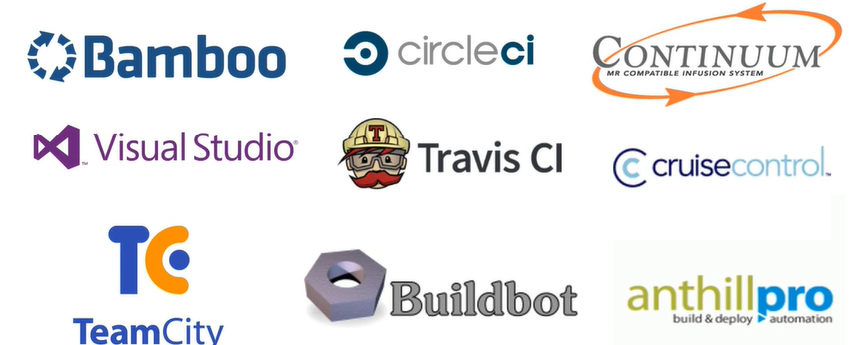
When Developers check-ins code to git repository, CI server fetches changes, integrates them and run build processes 🡺 produces a bundle of software build artifacts (versioned and stored in build repository) 🡺runs several automated tests like unit tests and integration tests to validate the build artifacts. 🡺 then it sends notifications to team about the success or failure of CI run.



Benefits:

1. By integrating code right before the QA cycle or before big release we are running a **high risk of last-minute failures**.
   1. By integrating codes changes coming from various team members and building a integrated source on a frequent basis, these risks are minimized.
2. Increased confidence for new features.
3. Better quality code: by frequent build testing, deployment, the bugs and errors were caught earlier in on the cycle by faster feedback loop
4. Ready to ship code
5. Systematic versioning.
6. Code quality and trend analysis. (gives all metrics)
7. Time to market.
8. Reduced cost.

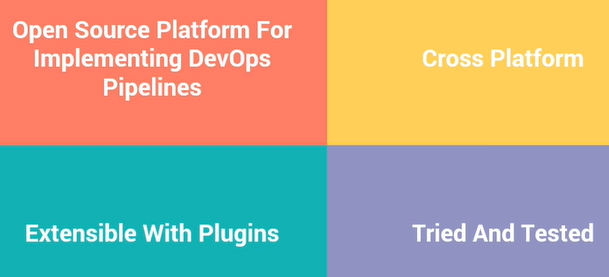
Tools for CI/CD



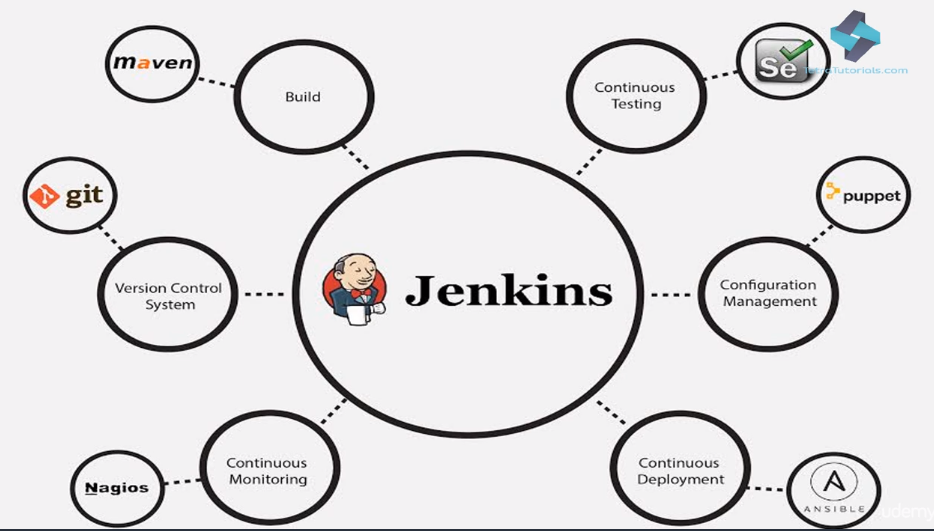
**JENKINS:** most popular tool for CI/CD pipeline.

It is a platform that can handle any kind of build, CI, CD or Devops task. Can integrate any number of testing and deployment technologies

* It is open source automation tool written in java with plug ins built for dev ops purposes.
* Can trigger a build for every change made in source code repository.
* Jenkins deploys these builds on a test server for testing.
* Concerned teams will gets notified about the build and test results.
* Finally, it deploys build application on production server.
* Thus, enabling teams to automate their build and releases process.
* **Powerful plugin ecosystem.** Also, can customize the plugins or can develop new plugins.
* **Increases Productivity of developer.**



How Jenkins work:



**Architecture:**

* Uses **master-slave** architecture to build distributed builds.
* Master slave communicate through TCP/IP protocol.
* Masters job is to pull the SCM repositories, Schedule build jobs and dispatch them to slave for actual execution. Then monitor the slaves possibly taking them online and offline.
* Primarily master job is to manage the builds and jobs across the whole cluster.
* Masters instance of Jenkins can also execute the build jobs directly, where as a slave instance is a java executable that runs on a remote machine.
* Slave Jenkins instance accept requests from the Jenkins master instance and execute the job and reports back the status.
* Additional slaves’ nodes can be added to system as the load on the system grows.

