[**JenkinsTutorial step by step**](http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html)

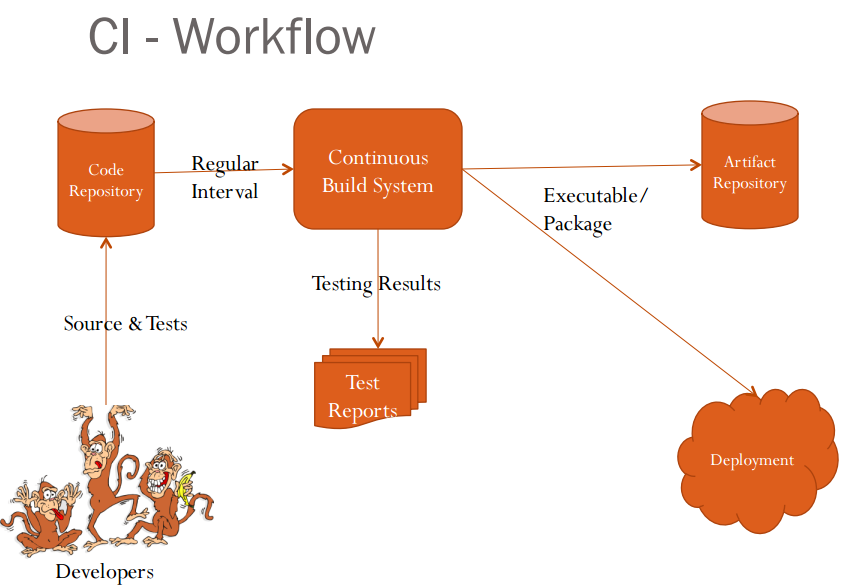
Standard

* **SHARE THIS:**
* [**FACEBOOK**](http://www.facebook.com/share.php?v=4&src=bm&u=http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html&t=JenkinsTutorial%20step%20by%20step)
* [**TWITTER**](http://twitter.com/home?status=JenkinsTutorial%20step%20by%20step%20--%20http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html)
* [**GOOGLE+**](https://plus.google.com/share?url=http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html)
* [**STUMBLE**](http://www.stumbleupon.com/submit?url=http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html&title=JenkinsTutorial%20step%20by%20step)
* [**DIGG**](http://digg.com/submit?phase=2&url=http://satyajohnny.blogspot.com/2015/11/jenkinstutorial-step-by-step.html&title=JenkinsTutorial%20step%20by%20step)

Jenkins – The Final Notes

1. **Introduction**

* Jenkins is an open source continuous integration tool written in Java.
* Jenkins provides continuous integration services for software development. I
* It is a server-based system running in a servlet container such as Apache Tomcat.
* Jenkins, a continuous build tool, automating the build, artifact management, and deployment processes



***CI – The tools***

**Code Repositories**

SVN, Mercurial, Git

**Continuous Build Systems**

Jenkins, Bamboo, Cruise Control

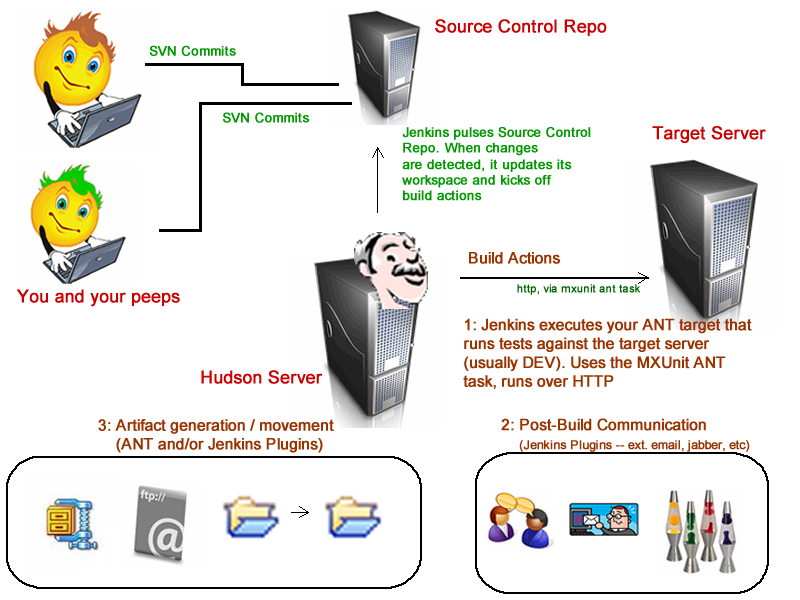
**Test Frameworks**

JUnit,Cucumber, CppUnit

**Artifact Repositories**

Nexus, Artifactory, Archiva

1. **How Jenkins Work**



1. ***Setup Level:***

Here we can choose what Options, Tools and plugins we can use with Jenkins

* Associating with a **version control server**
* **Triggering builds :**Polling, Periodic, Building based on other projects
* **Execution of shell scripts**, bash scripts, Ant targets, and Maven targets
* Artifact archival
* Publish **JUnit test results** and Javadocs
* Email notifications

1. ***Building***

***Once a project is successfully created in Jenkins, all future builds are automatic Building***

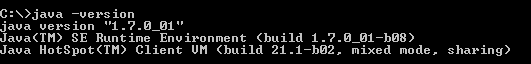
* Jenkins executes the build in an executer
* By default, Jenkins gives one executer per core on the build server
* Jenkins also has the concept of slave build servers
* Useful for building on different architectures
* Distribution of load

1. ***Reporting***

* Keeping track of build status: Last **success and failure**
* Unit test coverage Test result trending Findbugs, Checkstyle, PMD

1. **Types of Environments**
2. **Development**
3. **QA** ➔  only **Functional testing** of the system
4. **Integration Testing** ➔ Tests the system from **end to end**
5. User Acceptance Testing(**UAT**) ➔ **user will validate** the functionality over time
6. **Production** == Production
7. Production Parallel ➔ A parallel of production to replicate production issues
8. CERT ➔ CERT is Certification environment! It’s just where you **certify your product so that it can move to production**

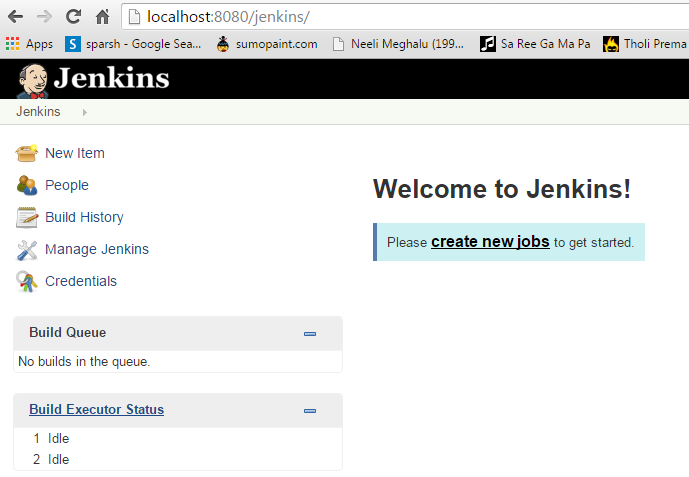
1. **Installing Jenkins**

* You can check the current version of Java that is installed on your machine by typing following command in command prompt.  
  

* Jenkins is distributed in the form of a bundled Java web application (a WAR file). You can download the latest version from the Jenkins website <http://jenkins-ci.org/>.

* Downloaded the WAR file, simply deploy the **Jenkins.war** file to your application server—with Tomcat. Extract and Place jenkins.war file in Tomcat’s webapps directory. Or Deploy using <http://localhost:8080/manager> Tomcat UI

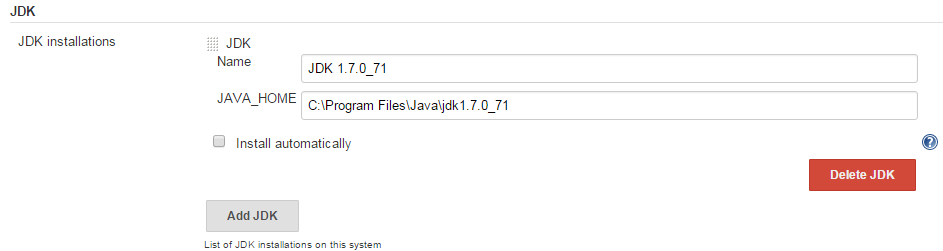
* On a default Tomcat installation you can access Jenkins in your web browser on <http://localhost:8080/jenkins>

[](http://localhost:8080/jenkins)

1. **Configure Jenkins**

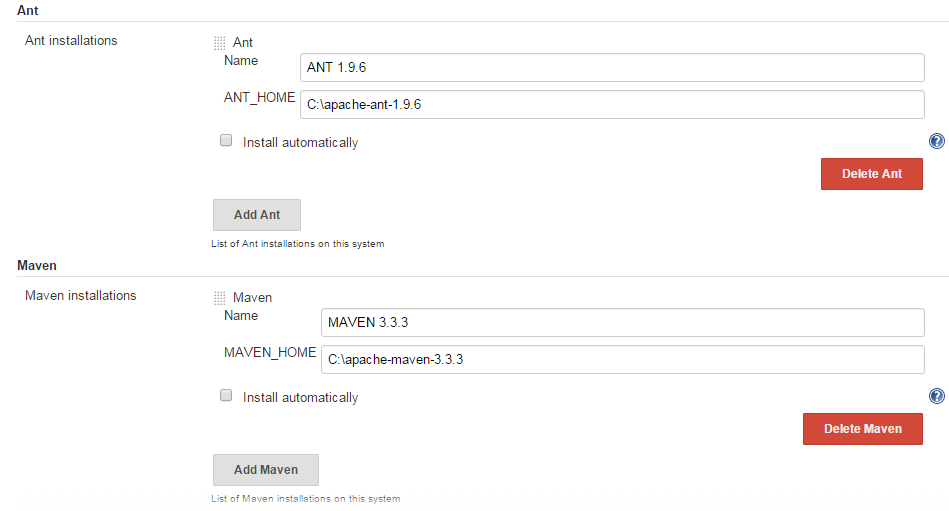
1. ***Configuring Java***

Manage Jenkins → Configure System → JDK Installations → Add JDK



1. ***Configuring ANT & MAVEN***

Manage Jenkins → Configure System → ANT & MAVEN



1. ***Secure Jenkins***

Manage Jenkins → Configure Secure Jenkins

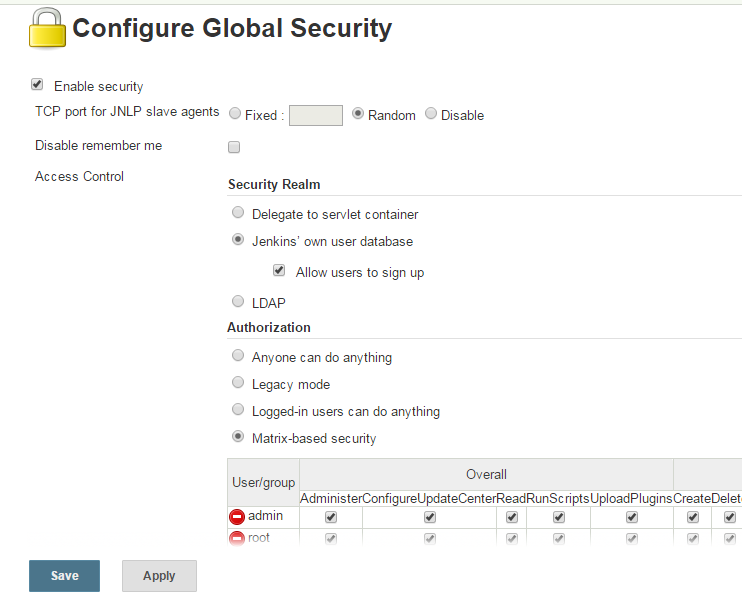
[Tick] Enable Security

[Select]   Jenkins own user database

[Tick] Allow users to Signup

[Select]   Matrix based Security

[add] User Group and give all the access



1. **Plug-in management**

***Manage Jenkins* → *Manager Plugins***   
link and search for the plugin you want to install. Select it from the list and select to install it and restart Jenkins.

You can manually restart Jenkins by adding *restart* as URL parameter.

1. **Setting up a Jenkins job**

The build of a project is handled via *jobs* in Jenkins. Select *New Item* from the menu

1. **Ant Project Configuration in Jenkins**

***a. Ant basics***

ANT stands for Another Neat Tool. It is a Java-based build tool from Apache.

**It will do**

* Compiling the code
* Packaging the binaries
* Deploying the binaries to the test server
* Testing the changes
* Copying the code from one location to another

Build.xml

Typically, Ant's build file, called **build.xml** should reside in the base directory

<?xml version="1.0"?>

  <project name="Hello World Project" default="info">

  <target name="info">

     <echo>Hello World - Welcome to Apache Ant!</echo>

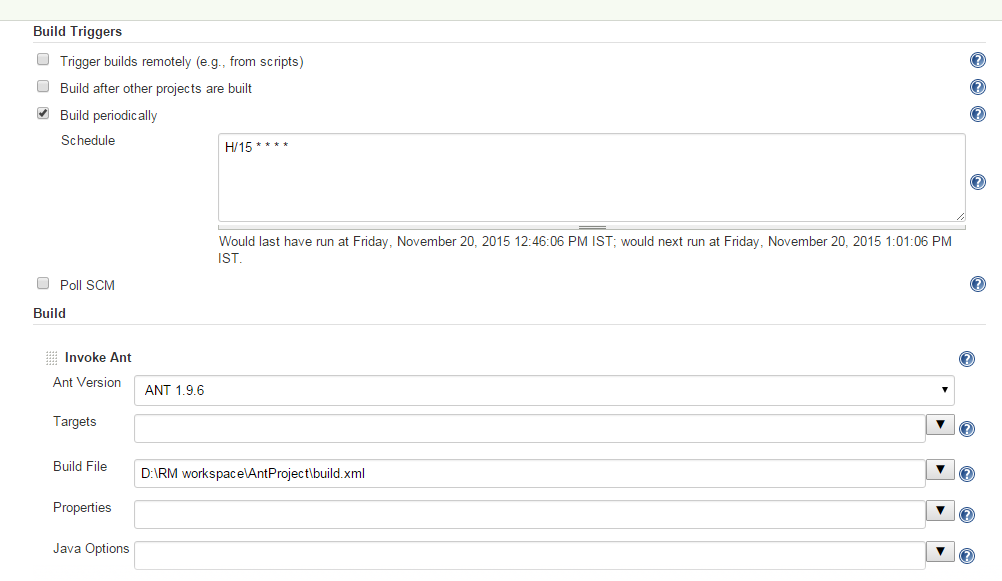
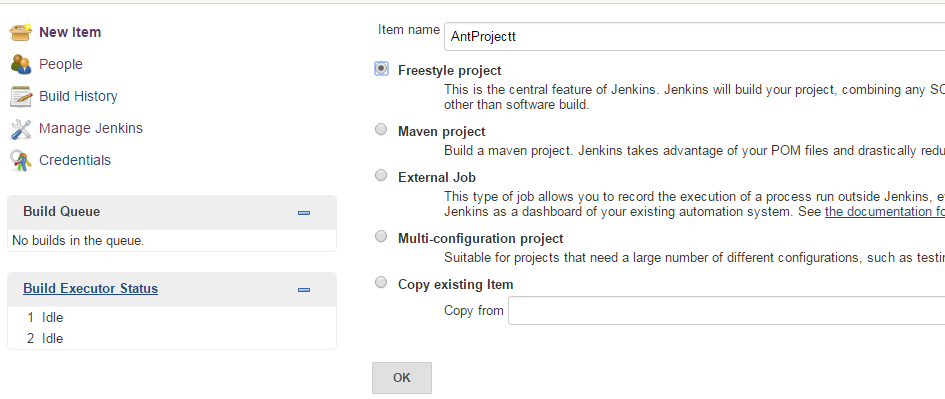
  </target>

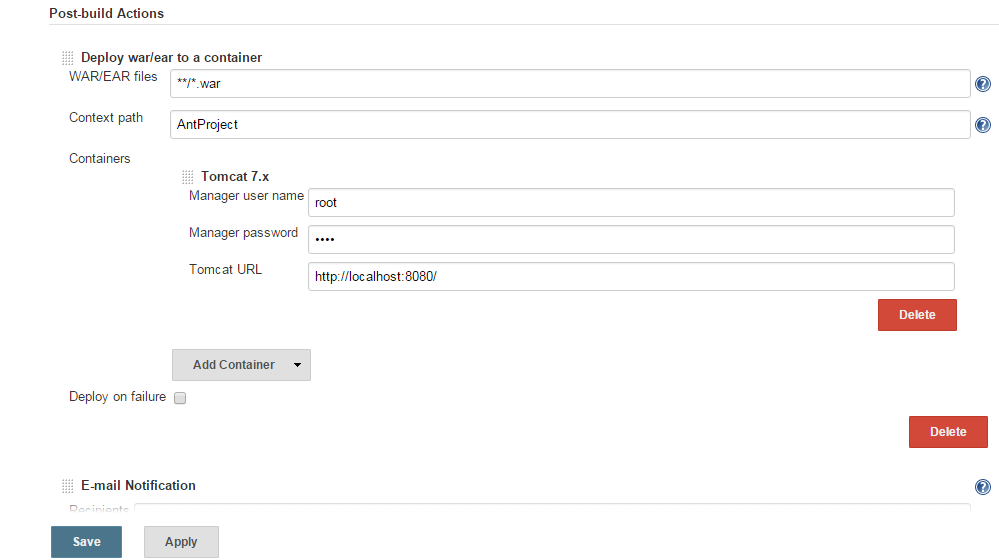
</project>

The XML element **project** has three attributes:

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| name | The Name of the project. (Optional) |
| default | The default target for the build script. A project may contain any number of targets. This attribute specifies which target should be considered as the default. (Mandatory) |
| basedir | The base directory (or) the root folder for the project. (Optional) |

***b. Ant – Jenkins Configuration***





Click on Build now it will automatically deploy war in Tomcat Server

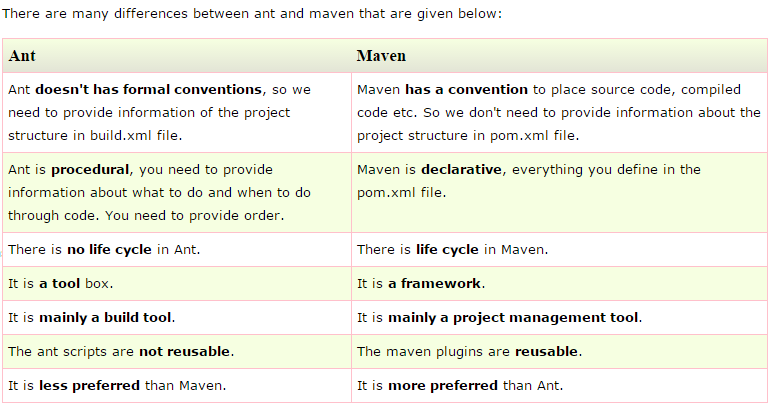
1. **Maven Project Configuration in Jenkins**

1. ***Maven basics***

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework.

**Maven provides developers ways to manage following:**

* Builds
* Documentation
* Reporting
* Dependencies
* SCMs
* Releases
* Distribution
* mailing list



**POM.xml**

* **POM** is an acronym for **Project Object Model**.
* The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.
* Maven reads the pom.xml file, then executes the goal.



**Elements of maven pom.xml file**

For creating the simple pom.xml file, you need to have following elements:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **project** | It is the root element of pom.xml file. |
| **modelVersion** | It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0. |
| **groupId** | It is the sub element of project. It specifies the id for the project group. |
| **artifactId** | It is the sub element of project. It specifies the id for the artifact (project).  An artifact is something that is either produced or used by a project.  Examples of artifacts produced by Maven for a project include:  JARs, source and binary distributions, and WARs. |
| **version** | It is the sub element of project. It specifies the version of the artifact under given group. |

**Maven pom.xml file with additional elements**

Here, we are going to add other elements in pom.xml file such as:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **packaging** | defines packaging type such as jar, war etc. |
| **name** | defines name of the maven project. |
| **url** | defines url of the project. |
| **dependencies** | defines dependencies for this project. |
| **dependency** | defines a dependency. It is used inside dependencies. |
| **scope** | defines scope for this maven project. It can be compile, provided, runtime, test and system. |



***b. Maven – Jenkins Configuration***

# [What is difference between Maven, ANT, Jenkins and Hudson?](http://javarevisited.blogspot.com/2015/01/difference-between-maven-ant-jenkins-and-hudson.html)

In short, though Maven and ANT are build tool but main difference is that maven also provides dependency management, standard project layout and project management. On difference between Maven, ANT and Jenkins, later is a continuous integration tool which is much more than build tool. You can setup your CI environment using Jenkins or Hudson and automatically build, test and deploy your Java project. Now last, main difference between Jenkins and Hudson, both are originate from same source code but one is closed source while other is open source. You can read the details in this article. Now let's start long story, what is difference between Maven and ANT or difference between Maven and Jenkins, or Maven vs Hudson are some of the frequently discussed questions among Java when developers. Well all four e.g.  ANT, Maven, Jenkins and Hudson are tools to help Java developers on build, unit testing, continues integration (CI) and project management. In this Java article we will explore each of Maven, ANT, Jenkins and Hudson to get basic idea of what they are, what benefit they offer and how they are used in Java JEE projects. Just to give you basic idea, ANT is a well known build tool, probably oldest among all. A build tool is used to create deliverable like [JAR file](http://javarevisited.blogspot.sg/2012/03/how-to-create-and-execute-jar-file-in.html) or [WAR file](http://javarevisited.blogspot.sg/2012/01/tomcat-javalangoutofmemoryerror-permgen.html) from Java source and resources for deployment.  
  
Maven came after ANT and offers much more than a build tool. *Main difference between ANT and Maven* is that In ANT you need to define every thing i.e. source directory, build directory, target directory etc while Maven adopts principle of *Convention over configuration*. Which means Maven has predefined project structure i.e. standard directory for source files, test files and resources. On the other hand, Jenkins and Hudson are Continues Integration tool, which gives you power to automate your build and deployment process. By using Jenkins or Hudson you can trigger build whenever developer commit code, to see if project is compiling fine, to run unit tests, to create build or even deploy in QA or production environment. Similarly you can have daily build, nightly build or weekly build process established in Jenkins or Hudson. In next section we will understand Maven, ANT, Jenkins and Hudson in more detail and understand difference between them.

## Maven vs ANT

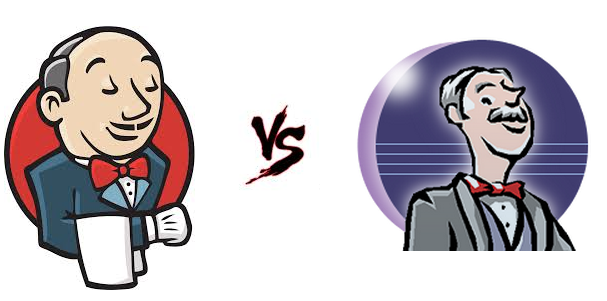
Ever since Maven has released, many Java programmer compare Maven and ANT, which is pretty natural. Even I have taken some time to adopt Maven because I was very used to [ANT build process](http://javarevisited.blogspot.sg/2010/10/ant-basics.html). Eventually I realized Maven offers more convenience as build tool than ANT and also enforces a common way to build Java applications among multiple projects, developers and organization. Before shifting to Maven, I had worked with very complex ANT scripts, in-fact every project I work has different build structure than other. In order to make a release, you need to spend considerable time to understand whole build process e.g. from where does libraries are coming, are they coming as JAR or project is building them during build time etc. Maven solves many of these problem by introducing standard convention and [better dependency management](http://javarevisited.blogspot.sg/2013/09/how-maven-find-dependency-jars-while-building-Java-project.html).  
  
If we need to compare Maven and ANT, following are my list of **difference between Maven and ANT** :

[](http://2.bp.blogspot.com/-JkmYfjjooh4/VMZOde4yXAI/AAAAAAAACaw/NmHwQs8Glss/s1600/Maven%2Bvs%2BANT.jpg)

1) One major difference between Maven and ANT is that Maven requires *less configuration* than ANT because it works on principle of Convention over configuration and assumes reasonable default e.g. java source file in ${basedir}/src/main/java, resources on ${basedir}/src/main/resources,JUnit test cases on ${basedir}/src/test/java etc. It also creates Java class files on ${basedir}/target/classes and JAR file on ${basedir}/target directory. On the other hand ANT requires all these directories supplied as configuration, usually in [ANT build file e.g. build.xml](http://javarevisited.blogspot.sg/2010/10/ant-tutorial-part-2.html).  
  
  
2) Another significant difference between Maven and ANT is *dependency Management*. Maven introduced concept of repository, which is a central place to store all libraries, JARs etc. Maven allows you to use central maven repository as well as local repository and automatically download dependency during build process. While ANT based project generally use ${lib} as directory to store dependencies. Changing and updating dependency is much easier in maven than ANT because you don't need to manually download dependency. Having an organization wide central repository also helps to remove redundancy across different projects.  
  
  
3) Third and most important difference between Maven and ANT is that, Maven offers a consistent and common interface to build Java projects. All you need to do is download the project and run mvn install to build it. Also by knowing maven conventions and looking at pom.xml, one can easily understand where source files are and what are project dependencies.  
  
  
4) Another technical difference between ANT and Maven is that ANT task doesn't have any lifecycle, you need to define targets and there dependencies. While Maven has lifecycle, which is invoked when you run commands like mvn install. Maven executes a series of steps as a result of this command to produce artifacts e.g. [JAR file](http://javarevisited.blogspot.sg/2012/10/5-ways-to-add-multiple-jar-to-classpath-java.html), which is end of life cycle.  
  
  
5) Maven also enforce a standard naming convention for artifacts defined using groupId, artifactId and version. Also one more difference between ANT and Maven is that, Maven is more than just a build tool, it act as project management tool and can generate reports etc, Though I have yet to use this feature.

## Maven vs Jenkins and Hudson

Jenkins and Hudson are Continuous Integration tool, commonly known as CI frameworks. They allows Java project to automate build and deployment process. By using CI tools like Hudson or Jenkins you can create hourly or daily builds automatically, you can also automatically [run your unit tests](http://javarevisited.blogspot.sg/2013/03/how-to-write-unit-test-in-java-eclipse-netbeans-example-run.html) and can deploy build to your QA or production environment. This helps if you are working in a large project and development team is spread across multiple locations. CI tolls can trigger build whenever developer commits code and check if project is compiling or not. By the way Jenkins and Hudson are from same source tree. Hudson was original project which was open source and supported by Sun. When Oracle bought Sun, it took control over Hudson name and logistic platform of Hudson. Many open source contributor was not comfortable with that which results in Jenkins, which is pretty much community driven.  
  
If we need to compare Maven and Jenkins or Hudson, following would be my list of differences between Maven and Hudson or Jenkins :

[](http://4.bp.blogspot.com/-dQUeiJmr09s/VMZOrQM7lwI/AAAAAAAACa4/B-YWD15BJjg/s1600/jenkins%2Bvs%2BHudson%2Bjava.png)

1) Main difference between Maven and Jenkins or Hudson is that Maven is a build tool which knows how to build project and Jenkins or Hudson provides trigger to build. You can control when to trigger build automatically using Jenkins or Hudson or any CI tool.  
  
  
2) Jenkins can use Maven as build tool. Also Jenkins or Hudson can do lot more than just building application, they can run all unit tests, deploy deliverable into configured servers etc. Essentially difference is that Maven is build tool while Jenkins of Hudson is a Continues Integration tool.  
  
  
That's all on d**ifference between Maven and ANT, Jenkins or Hudson**etc. Maven is pretty much standard of building Java projects now days and you can customize it to fit in your ANT based legacy environment. Some organization also has CI tools like Jenkins which can be used along with Maven to automated build, test and deploy process.

Read more: <http://javarevisited.blogspot.com/2015/01/difference-between-maven-ant-jenkins-and-hudson.html#ixzz4CDwFNa9G>