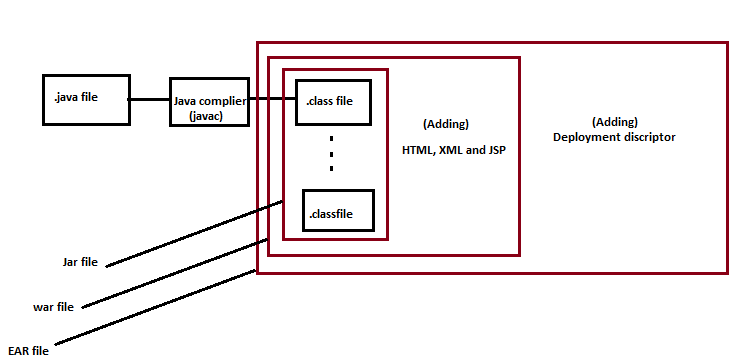
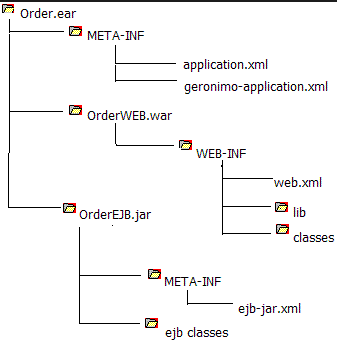
Build Management

Build Management is the process of assembling all the components of a software application into an installable software product. This process usually includes the following steps: The Preparing the Build Environment. The Gathering of the Source Code. The Labelling of the Source Code.

In the simple term we can say that build management is a process of creating EAR file that we can deploy/execute from Application server.





|  |  |
| --- | --- |
| Topic | Description |
| Jar file | Collection of .class file is called jar file. |
| War file | Jar file along with JSP, HTML and xml file called war file. |
| EAR file | war file along with deployment descriptor is called EAR file. |

We can automate EAR creation process with the help of ANT (Another nice tool) script with the help of build.xml file.

|  |  |
| --- | --- |
| Topic | Description |
| Copy tag | it copy the file from given location. |
| Replace tag | it replace a string into file with a new value. |

<replace file="${src}/index.html" token="@@@" value="wombat"/>

<copy todir="../backup/dir" <fileset dir="src\_dir"/> <globmapper from="\*" to="\*.bak"/> </copy>

Deployment descriptor

A deployment descriptor provides description of application that required by container to identify server-side resources.

Source Code repository- It is a place where we stored our code.

Service Management- Managing 3P.(People, process and policy)

Incident Management- any interruption of IT services is called incident.

Problem management – Identifying the Root Couse of incident and closing with a fix, so we may not again have problem again.

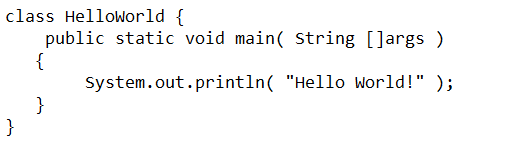
Operation management- managing OPS activity.

Apt-cache is a command-line tool used for searching apt packages on a Ubuntu or Debian based systems.

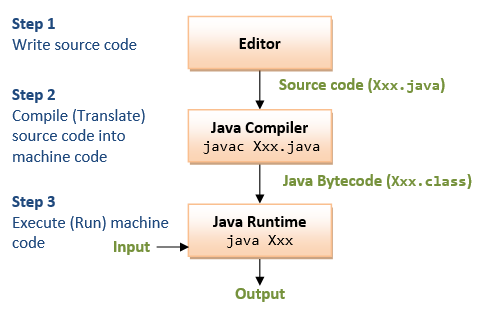
****



**Building Java Applications using Maven**



* JDK comes within built java compiler, which can be used to create the java compiled classes



If we have multiple .class file then we need to convert it to jar file. For that we would be using jar command as below.



* Build tools in Java help in building the jar/war files from the java code
* In this we would also like to run unit tests
* There was a popular tool called as Ant to perform these operations.
* In Ant we are supposed to write configuration files and in the configuration files we need to define the sequence of instructions that needs to be carried out.
* The instructions are generally written in build.xml file refer the below file



* for the sample java project with ant refer URL- <https://github.com/piona/ant-sample>
* When we write the java code, we use lot of code which is already. This is something which we call as dependency
* So now in this day, we need build tools to
  + build the java code
  + unit test the java code
  + manage the dependencies.

**Maven**

* Maven is a powerful build tool for Java Projects (Java Based Languages).
* Maven believes in Convention over Configuration
* The website of maven (<https://maven.apache.org/>
* Let’s use maven to build some java code
* Lab Setup
  + System 1: Create an ubuntu linux instance in any cloud/hypervisors and ensure it has internet connection
    - Install Open JDK8
  + sudo apt update
  + sudo apt install openjdk-8-jdk -y
    - JAVA is installed on the folder /usr/lib/jvm/java-8-openjdk-amd64 which is considered as JAVA HOME and we need to create an environment variable JAVA\_HOME representing home directory of java
  + export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64
    - Add the JAVA\_HOME to either /etc/environment or ~/.bashrc
    - Download the latest version of maven
  + cd /tmp
  + wget https://mirrors.estointernet.in/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
  + tar -xvzf apache-maven-3.8.1-bin.tar.gz
  + sudo mv apache-maven-3.8.1 /var/lib/maven-3.8.1
    - Now we need to create M2\_HOME environment variable which point to /var/lib/maven-3.8.1 and add /var/lib/maven-3.8.1/bin to path
    - Open ~/.bashrc and add the following lines to the end of the file
  + export JAVA\_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
  + export M2\_HOME="/var/lib/maven-3.8.1"
  + export PATH=$PATH:$JAVA\_HOME/bin:$M2\_HOME/bin
    - Now logout and login
  + mvn --version
    - Can we install maven latest version with JDK 11

Conventions over configurations

| **Item** | **Default** |
| --- | --- |
| source code | ${baseDir}/src/main/java |
| Resources | ${baseDir}/src/main/resources |
| Tests | ${baseDir}/src/test |
| Compile java classes | ${baseDir}/target/classes |
| Jar/War file | ${baseDir}/target |

**Maven POM**

* POM stands for Project Object Model.
* This is fundamental unit of work in Maven
* This contains information about the project and various configuration details used by Maven
* Goals:
  + compile: will compile the java code
  + test: will compile and run the test code
  + package: will test and create the jar/war file
  + install: will package and copy the package with pom file in local repository
  + deploy: will install and deploy the package to maven repository
  + clean : will remove the target folder
* In POM we will have the following information
  + project dependencies
  + plugins
  + goals
  + build profiles
  + project version

Elements of POM

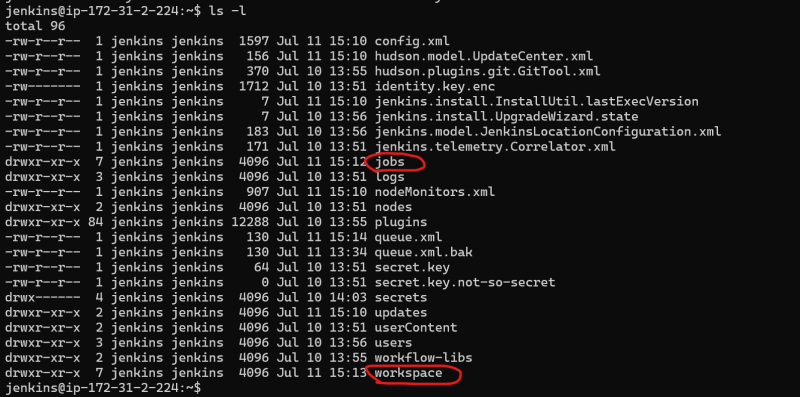
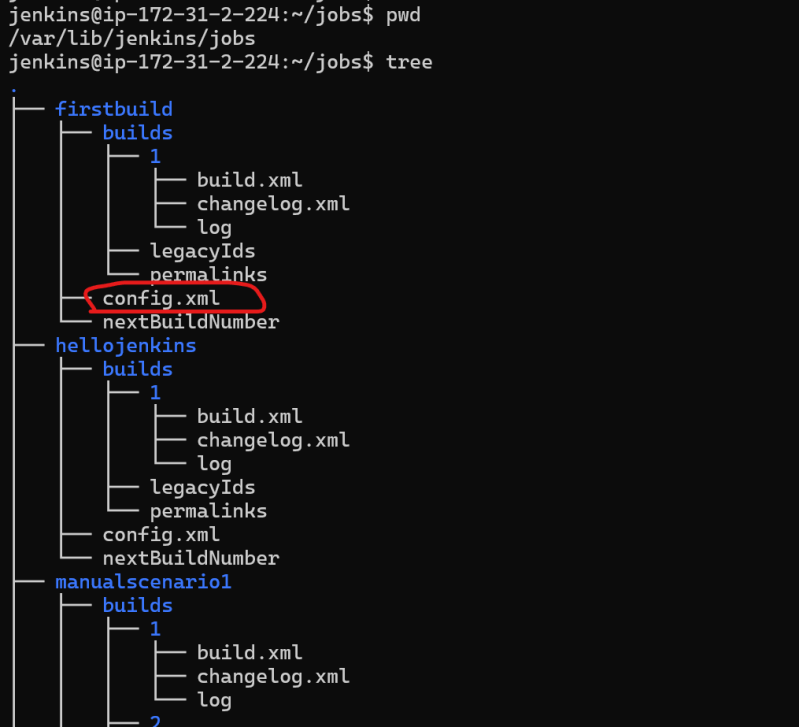
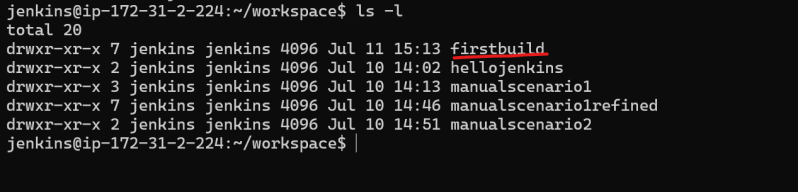
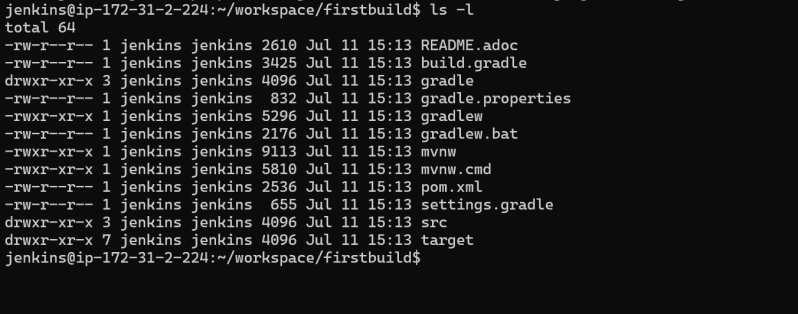
| **Element** | **Description** |
| --- | --- |
| modelVersion | refers to the schema version of maven and is generally 4.0.0.0 |
| groupId | This refers to the project which you are trying |
| artifactId | This refers to id of the project |
| version | This refers to Version of the project if version has SNAPSHOT it means the project is still under development |

* The build package will have the following format

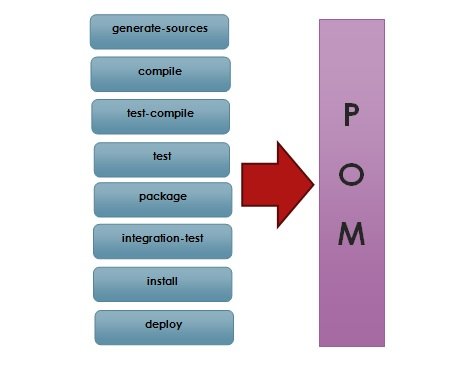
<artifactId>-<version>.<packagingformat>

* Super POM:
  + This is Maven’s default POM. All POMs inherit from this parent or default POM
  + Try executing

**Jenkins and Maven integration**

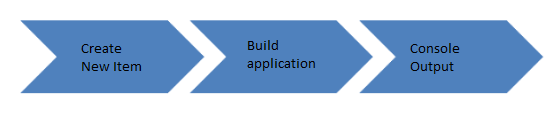
* From Jenkins let’s build a java project and show the jar files to user
* Execute mvn package from Execute shell after configuring git repository of java project Link: <https://github.com/neiljbrown/java11-examples>
* Now lets explore how Jenkins is storing the projects
* cd in to the home directory of jenkins (/var/lib/jenkins) as jenkins user and see the contents 
* Now get into jobs/your job name 
* The jenkins project created gets stored as xml file in /var/lib/jenkins/jobs/$job-name/config.xml
* Now lets cd into /var/lib/jenkins/workspace 
* Now lets cd into firstbuild 
* Jenkins when a project is created stores an xml file in the $JENKINS\_HOME/jobs/<your-job-name>/config.xml and when the project is executed all the work is done in the workspace folder $JENKINS\_HOME/workspace/<your-job-name>/

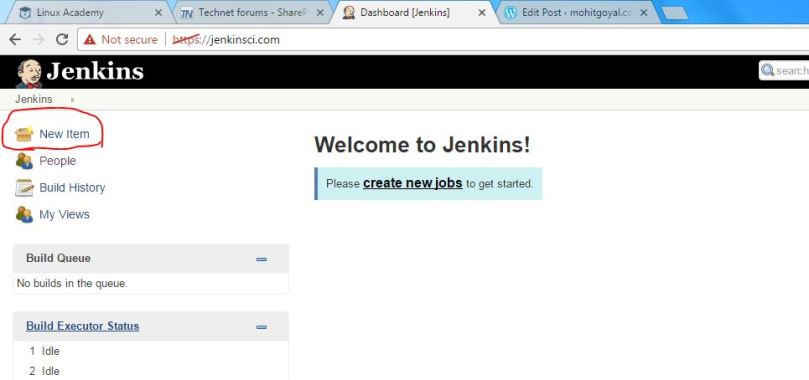
Maven build life cycle:



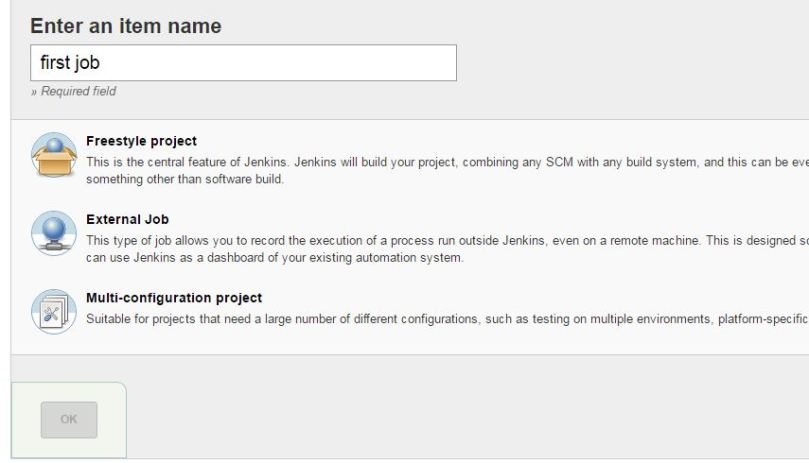
Creating sample job in Jenkin:

The types of actions you can perform in a Jenkins build step or post-build action are quite limited. There are many standard plugins available within a Jenkins Freestyle Project to help you overcome this problem

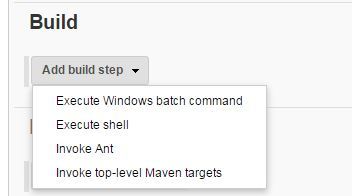
[](https://www.guru99.com/images/1/091318_0458_HowtoCreate1.png)

Select new item to start creating new jobs

If you are unable to see this icon, it means that you don’t have sufficient privileges. In the next window, type the name of the job such as first job and select job type as free style job and then click ok:

Select type of job

In the next window, we’ll leave default options unchanged. Move to build tab and select add build step:

 Add build step to the new job

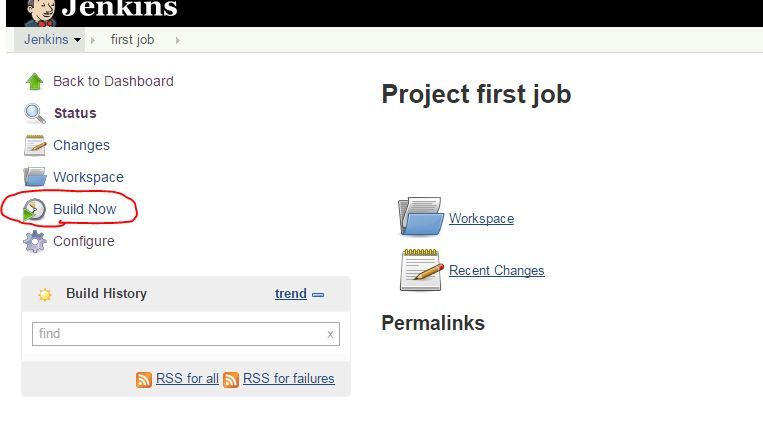
This list may vary a lot depending upon the plugins installed for Jenkins environment. For now, since this is a very basic job, we’ll just select ‘Execute shell’ and enter following command:

pwd  
ls -al

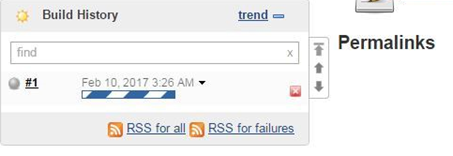
If you have installed jenkins on a window machine, select ‘Execute shell from the list, and then enter this command in the build step:

whoami date uptime

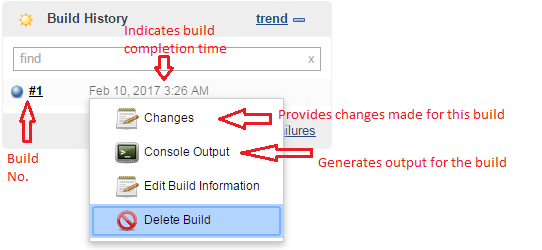
and click save. Once it’s done, click build now from left pane of the job:

Select build now from left pane of the job

Once you click, in the left pane only, under build history, you would notice that a new line with a number pops up along with a progress bar on it like this:



This means that, we have started first build for our job. Once the job is completed successfully, the icon ball color will turn from grey to blue and progress bar will go away. This build number keeps increment every time with the project build. Again, it can be customized to your choice. Once the build is complete, you would notice the time of build completion in front of build no. Along with it, there is a small black triangle which can be used to see various details with the build:



For now, we’ll click on the console output so that we can see logs generated for this build. It should be something like below:



In the ouput, we can see various useful details such as who started the build, what is the workspace for the build, what was the command executed by hudson for the build along with the project name, and offcourse, output related to our commands. In the last, we would also see the overall status of job as successfully completed or not.