**Jenkins -**

* + **Jenkins installation ways with explanation**

Jenkins as a service –

Download Jenkins for appropriate OS and install as a normal software and run the localhost:8080 on any Browser

Secret admin password copy from secretPassword and pest it at first time while Jenkins is startes

8080 is default port.

Jenkins on top of Tomcat:

Download Jenkins.war file from Jenkins site.

Copy Jenkins.war into apache webapps folder

Go to apache/bin and run startup.sh/bat file

Goto browser localhost:8080/Jenkins

Jenkins is fully up and running.

Admin password is find on apache running screen or in secrete file also u can find path

Temporarily run Jenkins –

Goto Start-cmd-path where Jenkins.war is available and

Run command java -jar Jenkins.war

If port no is need to change then run

java -jar Jenkins.war --httpPort=8085

* + **How to secure Jenkins**
  + **Plugin installation**

Manage Jenkins- manage plugins- Search for required plugin and install it.

* + **Mostly used plugins**
  + **How to restart Jenkins when jenkins as a service and on top of tomcat**

Jenkins as a service – localhost:8080\safeRestart (or restart)

Jenkins as application on top of Apache – restart Apache by shutdown.bat/.sh

* + **How to create job in Jenkins – all tabs description**
  + **Manage Jenkin0073**
  + **CI/CD pipeline/ Build pipeline**
  + **Which scripting used for Jenkins deployment automation**

Shell scripting

* + **How to enable/disable security**

Jenkins > Configure Global Security*--* and select the Enable Security checkbox

* + **How to login if password forgot**

1. Stop Jenkins via apache or shutdown.bat/.sh
2. Open config.xml (C:\Users\ma344818\.jenkins) of jenkins.
3. Find <useSecurity>true</useSecurity> change it with <useSecurity>false</useSecurity>
4. Start Jenkins via apache or start.bat/.sh
5. Now you are able to see the Jenkins home page without credential as anonymous user.
6. Go to manage jenkins 🡪Configure Global Security and enable security checkbox enable.
7. Now you will able to see the option manage user under manage Jenkins.
8. Click on settings icon of particular user and you can set new password for the user.
9. Save and apply.
10. Now login with username
    * **Database used in Jenkins**

You need to Enable security and set the security realm on the Configure Global Security page (see: [Standard Security Setup](https://wiki.jenkins.io/display/JENKINS/Standard+Security+Setup)) and choose the appropriate Authorization method (Security Realm).

1. Jenkins own database – This is suitable for smaller set up where you have no existing user database elsewhere.
2. LDAP (lightweight directory access protocol)
3. Delegate to servlet container
   * **Authorization securities**
4. Anyone can do anything

No authorization is performed. Everyone gets full control of Jenkins, including anonymous users who haven't signed on.

This is useful in situations where you run Jenkins in a trusted environment (such as a company intranet) and just want to use authentication for personalization support. In this way, if someone needs to make a quick change to Jenkins, they won't be forced to log in.

1. Legacy mode

Namely, if you have the "admin" role, you'll be granted full control over the system, and otherwise (including anonymous users) you'll only have the read access.

1. Logged in-user can do anything

In this mode, every logged-in user gets full control of Jenkins. The only user who won't have full control is anonymous user, who only gets read access.

This mode is useful to force users to log in before taking actions, so that you can keep record of who has done what. This setting can be also used in public-facing Jenkins, where you only allow trusted users to have user accounts.

1. Matrix based security --  [Matrix Authorization Strategy Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Matrix+Authorization+Strategy+Plugin)

In this scheme, you can configure who can do what by using a big table.

Each column represents a permission. Hover the mouse over the permission names to get more information about what they represent.

Each row represents a user or a group (often called 'role', depending on the security realm.) This includes a special user 'anonymous', which represents unauthenticated users, as well as 'authenticated', which represents all authenticated users (IOW, everyone except anonymous users.) Use the text box below the table to add new users/groups/roles to the table, and click the [x] icon to remove it from the table.

Permissions are additive. That is, if an user X is in group A, B, and C, then the permissions that this user actually has are the union of all permissions given to X, A, B, C, and anonymous.

1. Project based matrix authorization strategy --  [Matrix Authorization Strategy Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Matrix+Authorization+Strategy+Plugin)

This mode is an extension to "Matrix-based security" that allows additional ACL matrix to be defined for each project separately (which is done on the job configuration screen)

This allows you to say things like "Joe can access project A, B, and C but he can't see D." See the help of "Matrix-based security" for the concept of matrix-based security in general.

ACLs are additive, so the access rights granted below will be effective for all the projects

* + **Cron tab**

Command to be executed for cron tab - \* \* \* \* \*

1. \* ===== min (0 – 59)
2. \* ===== hour (0 - 23)
3. \* ===== day of month (1 - 12)
4. \* ===== month (1 - 12)
5. \* ===== Day of week (0 – 7 == 0 and 7 points to sunday)

|  |  |
| --- | --- |
| **\*** | any value |
| **,** | value list separator |
| **-** | range of values |
| **/** | step values |
| **@yearly** | (non-standard) |
| **@annually** | (non-standard) |
| **@monthly** | (non-standard) |
| **@weekly** | (non-standard) |
| **@daily** | (non-standard) |
| **@hourly** | (non-standard) |
| **@reboot** | (non-standard) |

E.g

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| min | hour | day/month | month | day/week | Execution time |
| 30 | 0 | 1 | 1,6,12 | \* | — 00:30 Hrs  on 1st of Jan, June & Dec. |
|  | | | | | |
| 0 | 20 | \* | 10 | 1-5 | –8.00 PM every weekday (Mon-Fri) only in Oct. |
|  | | | | | |
| 0 | 0 | 1,10,15 | \* | \* | — midnight on 1st ,10th & 15th of month |
|  | | | | | |
| 5,10 | 0 | 10 | \* | 1 | — At 12.05,12.10 every Monday & on 10th of every month |

* + **what is syntax if you want to install jenkins on different port**
    - java -jar jenkins.war --httpPort=8088

1. **What is ANT.**

ANT (Another Neat Tool)is a Build Tool based on Java.

Tasks –

Compiling java code into byte code

Place this byte code into pkg

Creation of .jar/war/ear files.

Deployment to production env

1. **ANT Functionality**

Open source under Apache so that we can download and update source code.

Can be integrated with any editor or development env easily.

Ant uses XML build files which make its development easy.

1. **Explain ANT with example from start.**

**Steps**

* 1. Download apache ANT
  2. Set **JAVA\_HOME, ANT\_HOME** and edit **PATH** by appending **%JAVA\_HOME%/bin**; and **%ANT\_HOME%/bin**
  3. Write “Hello world” build.xml

<project name=”HelloWorld” default=”compiler” basedir=”.”>

<target name=”compiler”>

<mkdir dir = “Helloworldclassfiles”>

<javac srcdir=”Dirhelloworld” destdir=”Helloworldclassfiles”>

</target>

</project>

* + **If Build.xml is not available in dir and is saved with different name**
  + **Tags**
  + **Build.xml file structure**

<?xml version="1.0"?>

<project name="Ant-Test" default="main" basedir=".">

*<!-- Sets variables which can later be used. -->*

*<!-- The value of a property is accessed via ${} -->*

<property name="src.dir" location="src" />

<property name="build.dir" location="bin" />

<property name="dist.dir" location="dist" />

<property name="docs.dir" location="docs" />

*<!-- Deletes the existing build, docs and dist directory-->*

<target name="clean">

<delete dir="${build.dir}" />

<delete dir="${docs.dir}" />

<delete dir="${dist.dir}" />

</target>

*<!-- Creates the build, docs and dist directory-->*

<target name="makedir">

<mkdir dir="${build.dir}" />

<mkdir dir="${docs.dir}" />

<mkdir dir="${dist.dir}" />

</target>

*<!-- Compiles the java code (including the usage of library for JUnit -->*

<target name="compile" depends="clean, makedir">

<javac srcdir="${src.dir}" destdir="${build.dir}">

</javac>

</target>

*<!-- Creates Javadoc -->*

<target name="docs" depends="compile">

<javadoc packagenames="src" sourcepath="${src.dir}" destdir="${docs.dir}">

*<!-- Define which files / directory should get included, we include all -->*

<fileset dir="${src.dir}">

<include name="\*\*" />

</fileset>

</javadoc>

</target>

*<!--Creates the deployable jar file -->*

<target name="jar" depends="compile">

<jar destfile="${dist.dir}\de.vogella.build.test.ant.jar" basedir="${build.dir}">

<manifest>

<attribute name="Main-Class" value="test.Main" />

</manifest>

</jar>

</target>

<target name="main" depends="compile, jar, docs">

<description>Main target</description>

</target>

</project>

* + - **Project tag**
    - **Target**
    - **Depends**
    - **Task**
    - **Property file etc**
  + **How to run build.xml**
  + **How to integrate ant with Jenkins**
  + **Troubleshooting of build errors in Jenkins**

1. **Difference between ant and maven**
2. **Maven basics**

**What is POM.xml**

**What can maven does**

**POM.xml file structure**