**Jenkins -**

* + **What is jenkins**

Jenkins is a open source written in java with plugins built for CI purpose.

Jenkins can be easily integrate with number of different tools.

Free of cost and easy to install and handle.

Java based so platform independent.

* + **What is Continuous Integration**

CI is the process of integrating the code which is developed by the developers. A developer required to commit the code into repository several time in a day or frequently into the repo.

Every commit made build.

This allows the team to detect problems early.

So that, its less time consumes.

Apart from this, depending upon the CI tool, there are several other functionalities are provided like deploy the application into test server.

* + **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

Through CLI - 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

On Linux Platform:

The complete process to install Jenkins can be summarized in five steps:

1. Install Java Version 8 – Jenkins is a Java based application, hence Java is a must.

**sudo yum install java-1.8.0-openjdk**

1. Install Apache Tomcat Version 9 – Tomcat is required to deploy Jenkins war file.

**sudo yum install wget**

**wget** <https://archive.apache.org/dist/tomcat/tomcat9/v9.0.0.M10/bin/apache-tomcat-9.0.0.M10.tar.gz>

**tar xzf apache-tomat-9.0.0.M10.tar.gz**

**mv apache-tomcat-9.0.0.M10 tomcat9**

**gedit /home/edureka/tomcat9/conf/tomcat-users.xml**

**cd tomcat9**

**./bin/startup.sh**

1. Download Jenkins war File – This war is required to install Jenkins.

**Cd**

**wget**<http://updates.jenkins-ci.org/download/war/2.7.3/jenkins.war>

**mv jenkins.war /home/edureka/tomcat9/webapps**

1. Deploy Jenkins war File – Jenkins war file needs to be deployed using Tomcat to run Jenkins.

* open your browser and access localhost:8080 again. Now click on the Manager App.
* Click Jenkins application to run.

1. Install Suggested Plugins – Install a list of plugins suggested by Jenkins.

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

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

* + **Build using ANT and git**

1. Create new job
2. Configure
3. Select GitHub in build task
   * **Mostly used plugins**
4. GitHub Plugin
5. Ant Plugin
6. Deploy to container plugin
7. Email extension plugin
8. Delivery pipeline plugin
9. Role strategy plugin
   * **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**

New Item

Job name

Select project type – freestyle provides more flexibility

Ok

Job configuration – having 6 sections

1. General

In general section

Description – you can add description of the project

GitHub Project – can add github project URL

Parameters – can add parameters for the project

Disable project if you want.

Execute concurrent build if necessory

1. SCM

Add repository details

e.g. Git –(Git plugin required)

Put repository URL

1. Build triggers – github hooks option

Trigger build remotely

Build after other project build

Build periodically

Poll SCM

1. Build environment

Specify the build environment like

Delete workspace before build starts

Abort build if it stuck

Add timestamp to console output

With ANT

1. Build

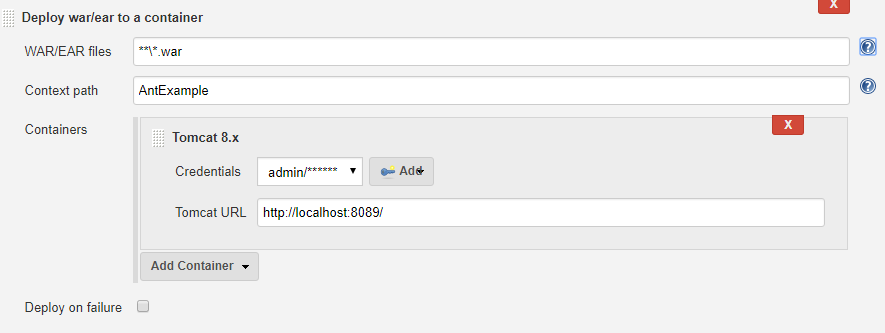
Add build steps like

Execute window batch command / Shell

Invoke ANT – (ant version and default target speacify in Buil.xml in adv. Build file name in case if file name is not as buil.xml)/Gradle/Maven target etc – need plugin

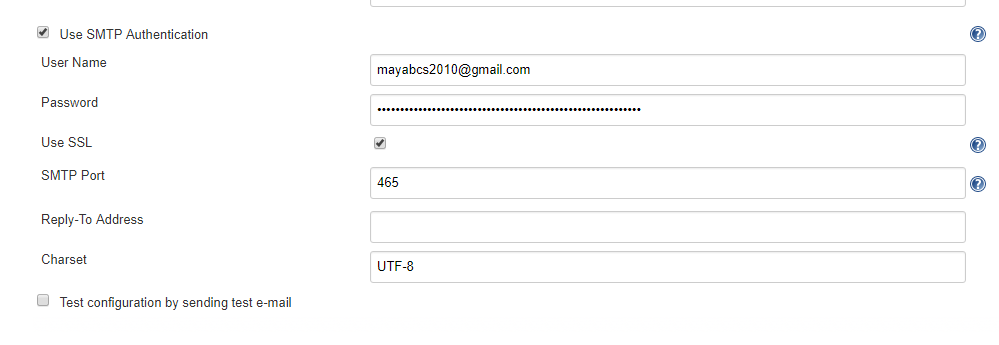
1. Post-build action

Deploy ear/war file to container – Deploy to container plugin



Email notification – Jenkins mailer plugin

Manage Jenkins 🡪 configure system

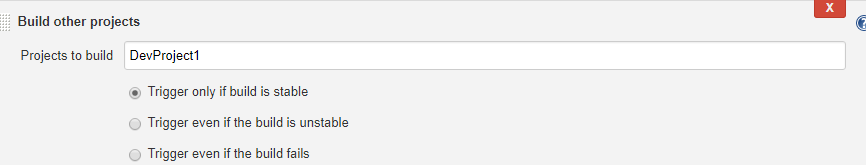


If configured, Jenkins will send out an e-mail to the specified recipients when a certain important event occurs.

1. Every failed build triggers a new e-mail.
2. A successful build after a failed (or unstable) build triggers a new e-mail, indicating that a crisis is over.
3. An unstable build after a successful build triggers a new e-mail, indicating that there's a regression.
4. Unless configured, every unstable build triggers a new e-mail, indicating that regression is still there.

Delete workspace when builds is done

Build other project



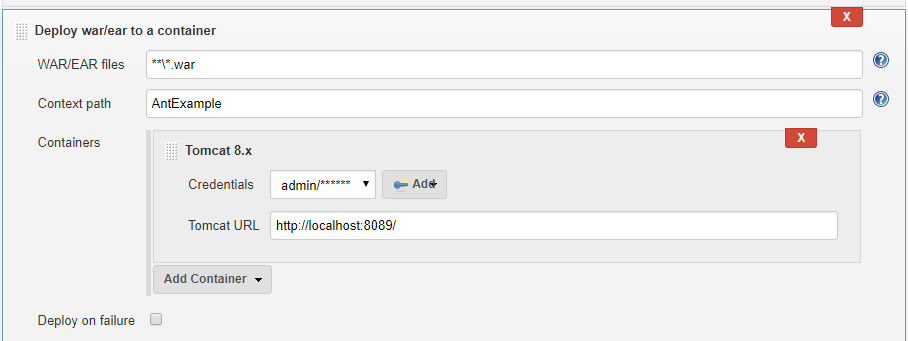
* + **Manage Jenkin**
  + **CI/CD pipeline/ Build pipeline**

How to setup Delivery Pipeline in Jenkins?

Follow along step by step

1. Chain required jobs in sequence Add upstream/downstream jobs
2. Install Delivery Pipeline Plugin
3. Add Delivery Pipeline View configure the view
4. Run and Validate
   * **Post Build Action**
5. **Deployment :**

* Install deploy to container plugin
* Configure job (which is already build successfully and created .war file)
* Goto post build action – Deploy to container
* Need another instance of tomcat
* Need to change port numbers in server.xml and role/ user in tomcat-users.xml.
* Run another instance of tomcat i.e localhost:8089
* Run the project and verify result through manager app or append .war file name in front of URL e.g **localhost:8089/AntExample**
  + - Reference: <https://www.jdev.it/deploying-your-war-file-from-jenkins-to-tomcat/>



1. **Email notification**

* Install Email-extension Plugin
  + **Which scripting used for Jenkins deployment automation**

Shell scripting

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

Manage 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

1. Role based strategy - Enables defining authorizations using a role-based strategy. Once the strategy is enabled, it can be configured via a separate page in the Manage Jenkins window.

**Jenkins authentication and authorization**

1. Create new users
2. Configure users – after login as user1 you can see top town arrow near logged in user click on that and click on configure
3. Create and manage user roles will visible once download plugin

Role based authorization Strategy Plugin - download - restart Jenkins is must

1. Login as admin - Manage Jenkins - Configure Global Security – check enable security then go to Authorization – select Role Based Strategy

Once you have select, try to login with the user newly create, will get error as Access denied.

So now--

1. Login as admin – manage Jenkins – manage and assign roles (will get option only after select Role Based Strategy in above step) –

manage roles –

Three option you will find here you can add roles under these options

1. Global roles – authorization and access at global level (generally give **overall - Read** and **view -all** access)
2. Project roles -
3. Slave roles

Assign roles –

Assign roles to the users by project (Item role) or globally.

Create Roles and Assign roles to users

1. Validate authorization and authentication are working properly
   * **Cron tab**

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

1. \* ===== min (0 – 59)
2. \* ===== hour (0 - 23)
3. \* ===== day of month (1 - 31)
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**