Jenkins

(Basic)

By

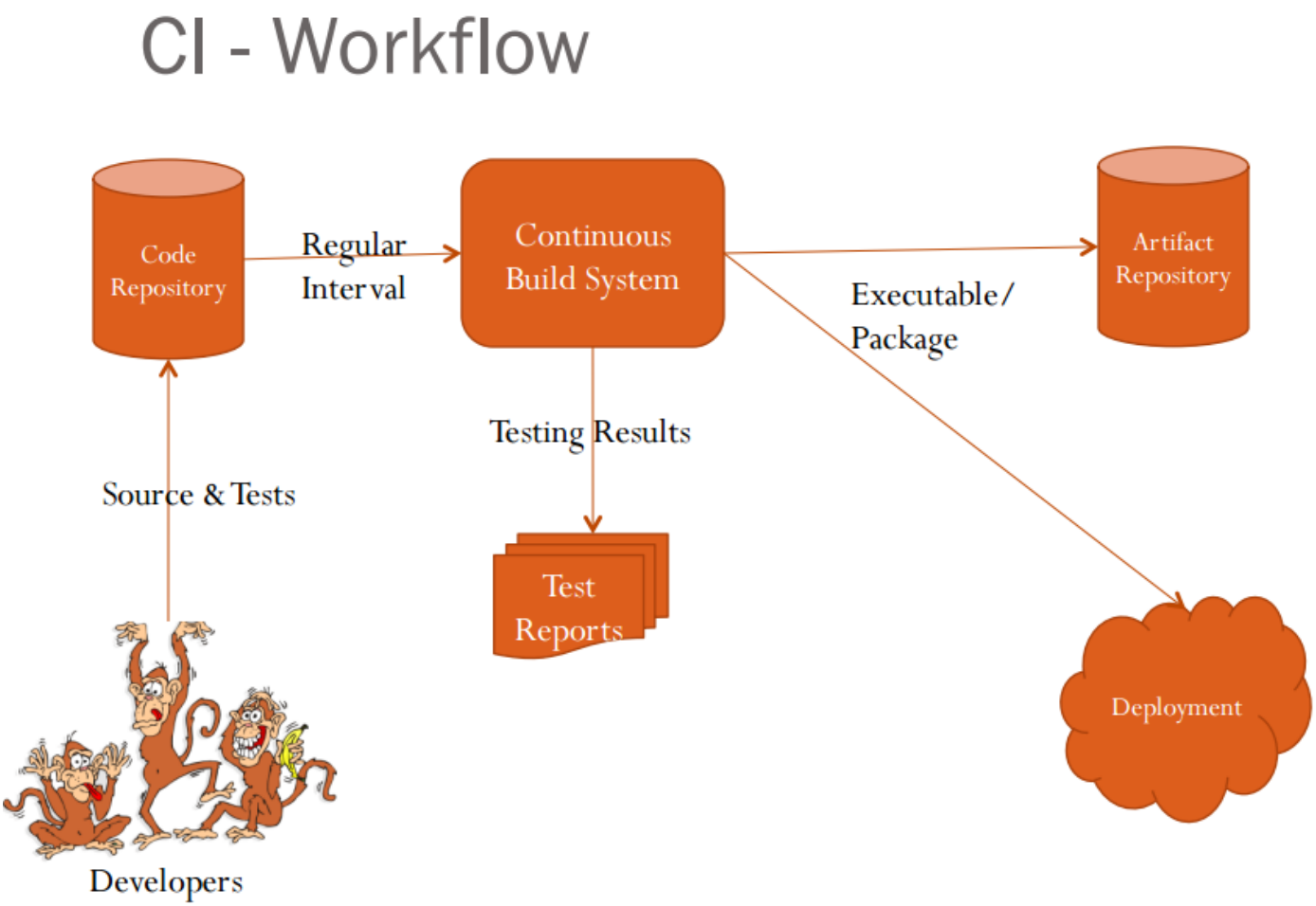
Satya Kaveti

http://ameerpetmaterials.blogspot.in/

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

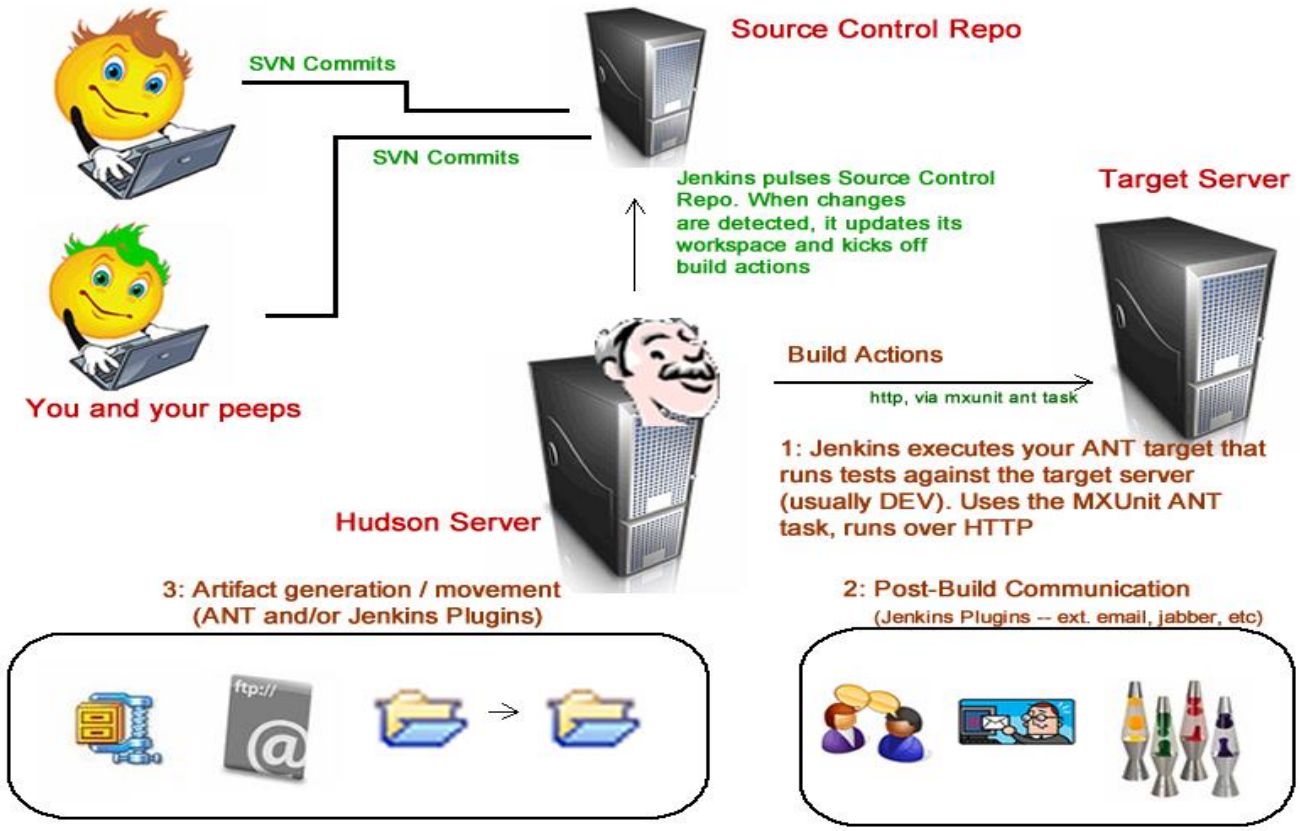
Test Frameworks

JUnit,Cucumber, CppUnit

Artifact Repositories

Nexus, Artifactory, Archiva

2. How Jenkins Work



i. 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

ii. 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

iii. Reporting

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

3. Types of Environments

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

4. Installing Jenkins

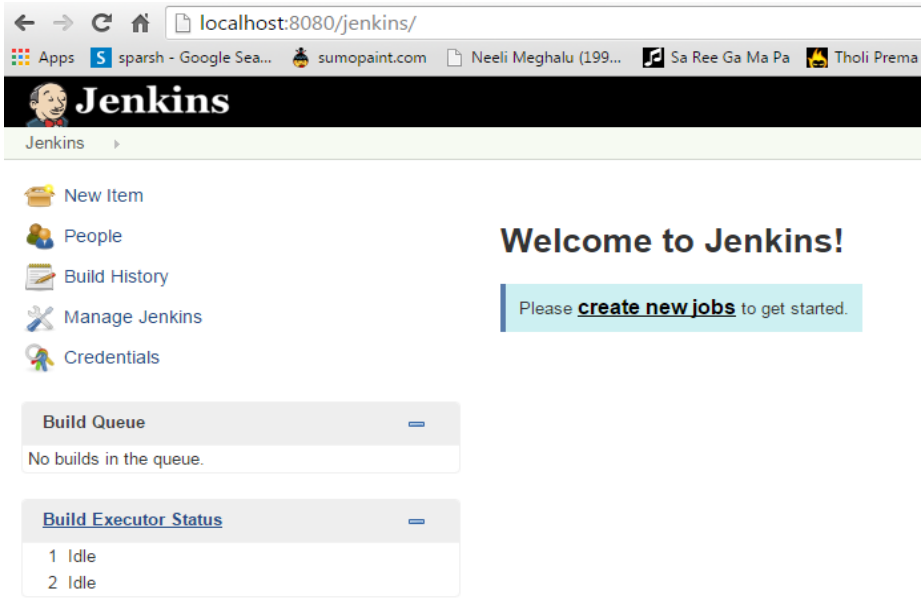
- You can check the current version of Java that is installed on your machine by typing following command in command prompt.

```
C:\>java -version
java version "1.7.0_01"
Java(TM) SE Runtime Environment (build 1.7.0_01-b08)
Java HotSpot(TM) Client VM (build 21.1-b02, mixed mode, sharing)
```

- 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

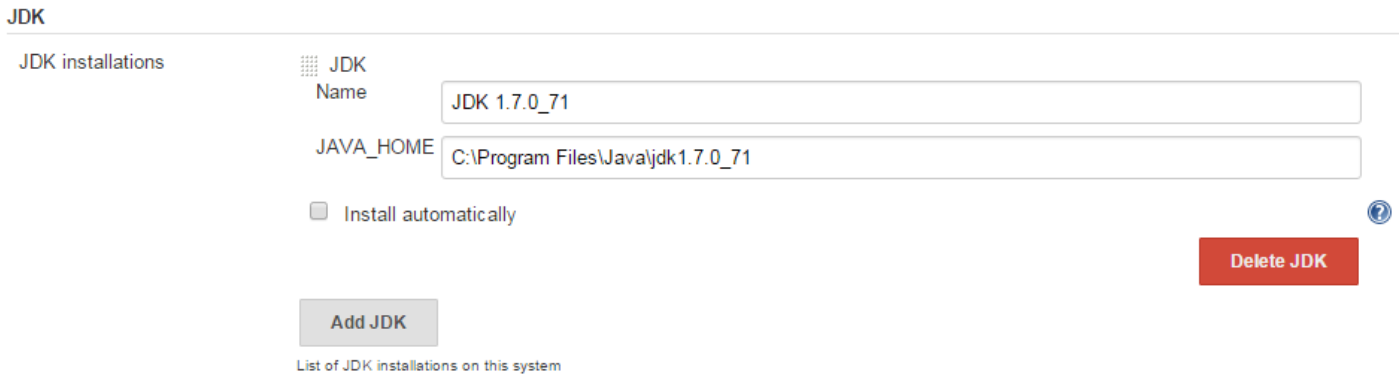
- On a default Tomcat installation you can access Jenkins in your web browser on http://localhost:8080/jenkins



5. Configure Jenkins

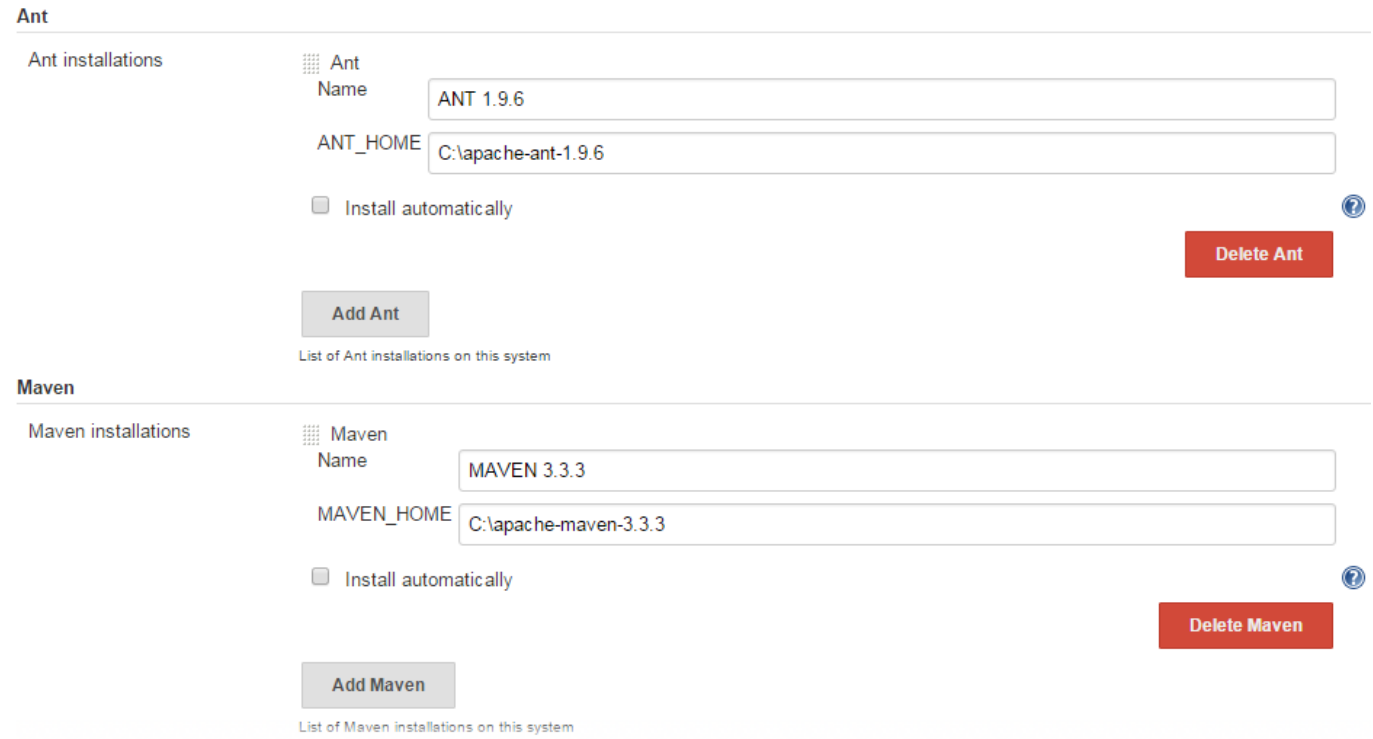
I. Configuring Java

Manage Jenkins → `Configure System` → `JDK Installations` → `Add JDK`



II. Configuring ANT & MAVEN

Manage Jenkins → Configure System → ANT & MAVEN



III. Secure Jenkins

Manage Jenkins → Configure Secure Jenkins

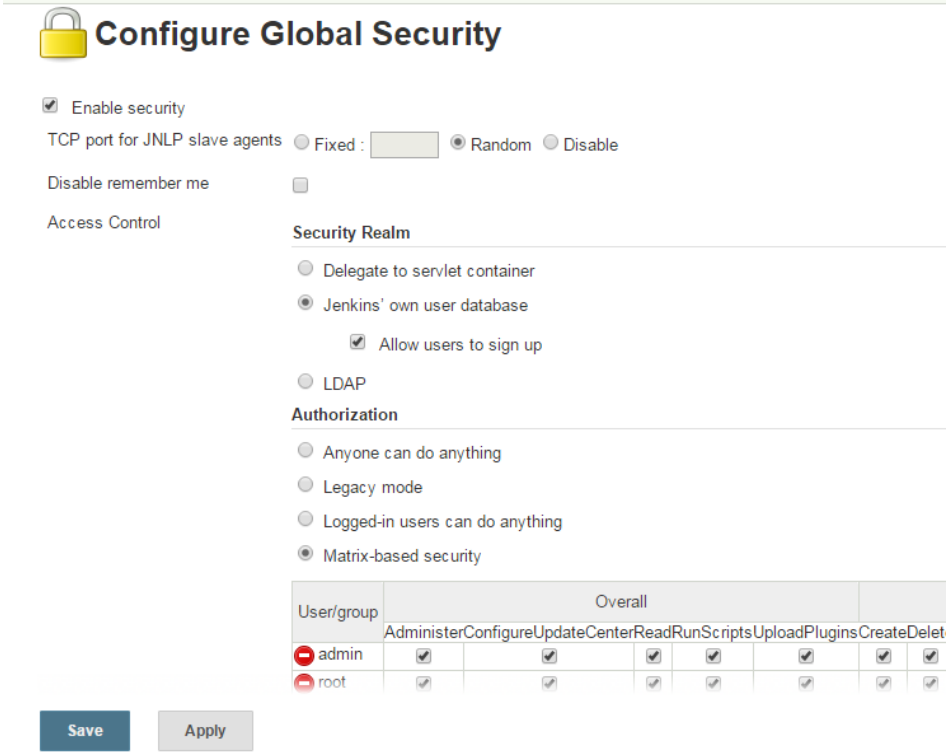
[Tick] Enable Security

[Select] Jenkins own user database

[Tick] Allow users to Signup

[Select] Matrix based Security

[add] User Group and give all the access



6. 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.

7. Setting up a Jenkins job

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

8. 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

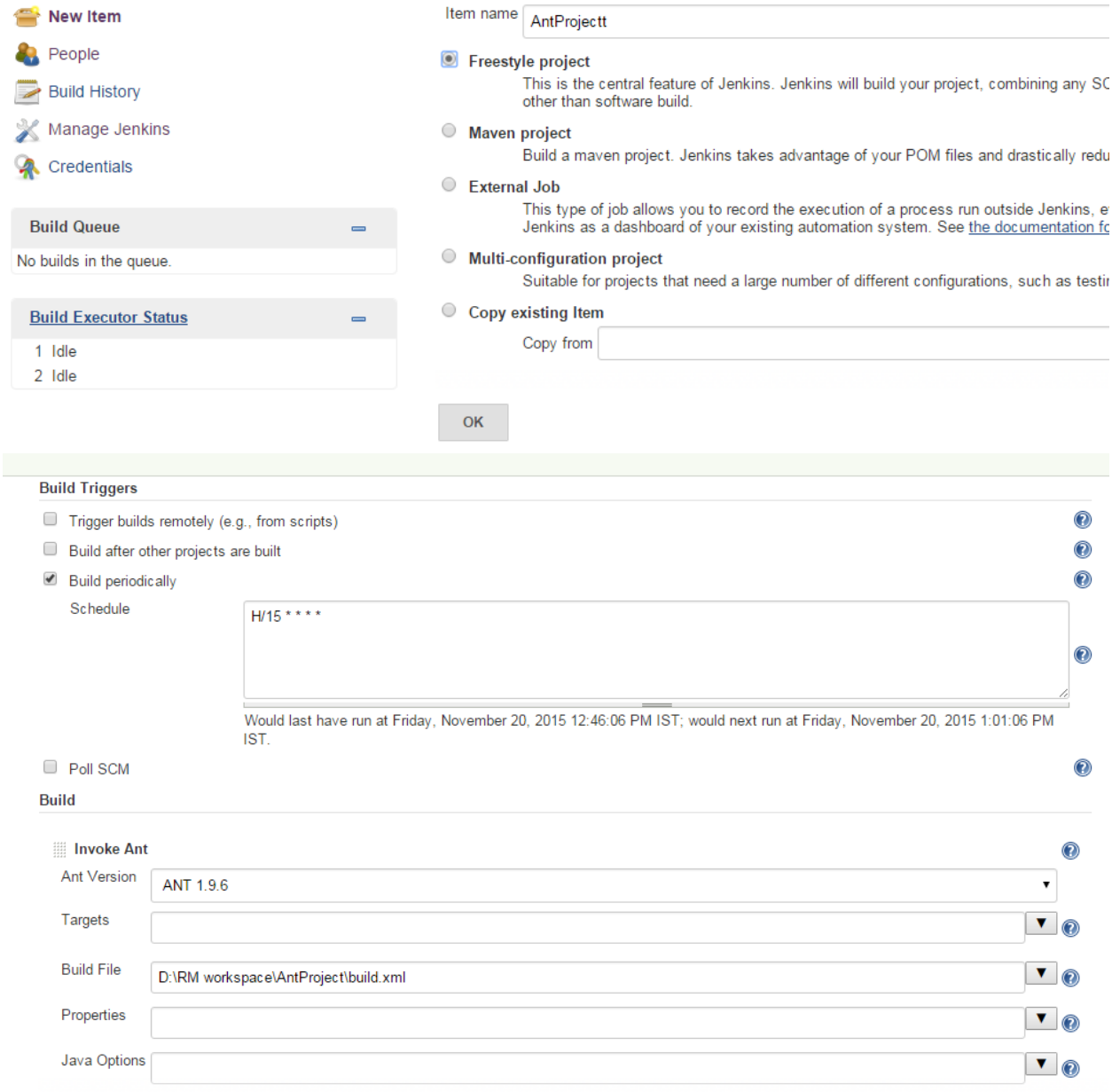
Typically, Ant's build file, called **build.xml** should reside in the base directory

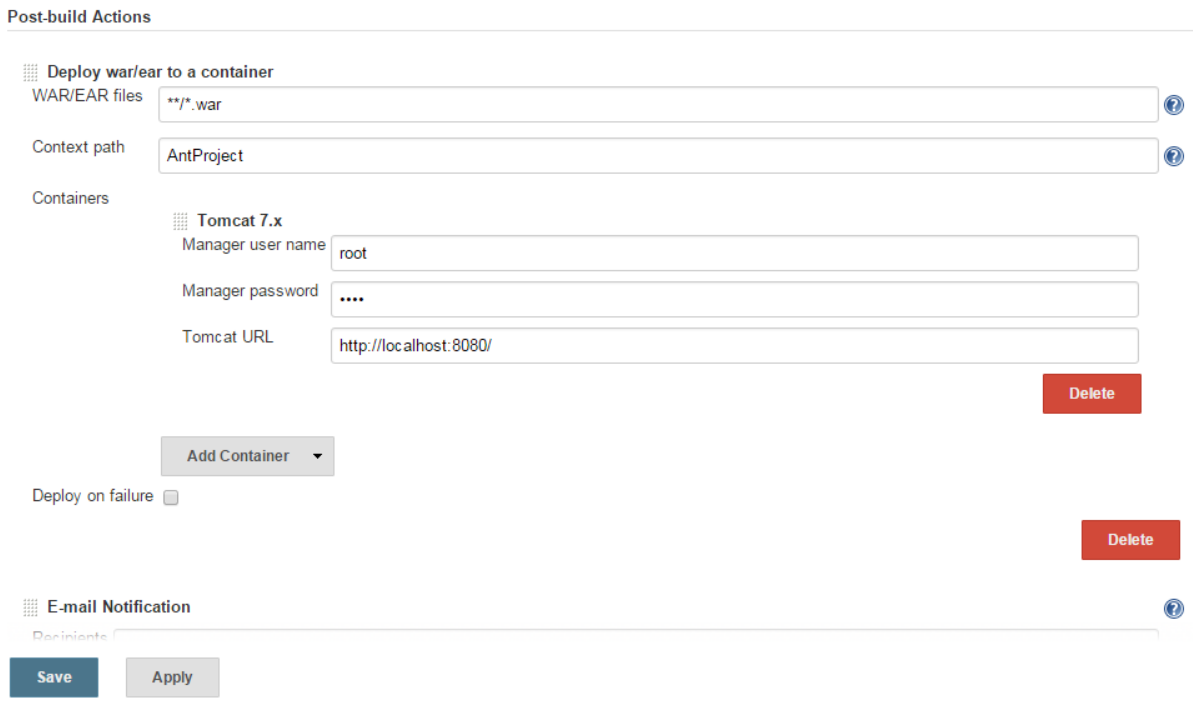
```
<?xml version="1.0"?>
   <project name="Hello World Project" default="info">

   <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

9. Maven Project Configuration in Jenkins

a. 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

There are many differences between ant and maven that are given below:

Ant	Maven
Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file.	Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order.	Maven is **declarative**, everything you define in the pom.xml file.
There is **no life cycle** in Ant.	There is **life cycle** in Maven.
It is **a tool** box.	It is **a framework**.
It is **mainly a build tool**.	It is **mainly a project management tool**.
The ant scripts are **not reusable**.	The maven plugins are **reusable**.
It is **less preferred** than Maven.	It is **more preferred** than Ant.

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.

File: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javatpoint.application1</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>

</project>
```

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.

```
<modelVersion>4.0.0</modelVersion>

<groupId>com.javatpoint.application1</groupId>
<artifactId>my-application1</artifactId>
<version>1.0</version>
<packaging>jar</packaging>

<name>Maven Quick Start Archetype</name>
<url>http://maven.apache.org</url>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>

</project>
```

b. Maven – Jenkins Configuration

Refer: http://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html