


Hudson Plugin Categories and Usage Analysis

Winston Prakash

Jan 2013





The slides in this document present Hudson Plugin Categories based on their importance in a Continuous Integration System and usage by community.

See also

<http://wiki.hudson-ci.org/display/HUDSON/All+Plugins+by+Topic>

Successful Agile Team

The three main concepts,

- Test Driven Development
- Continuous Integration
- Continuous Delivery or Deployment

are the solid supporting pillars of a Successful Agile Team.

In a **Test Driven Development** build pipeline, **Continuous Integration** is the first step and the end result is the **Continuous Delivery**.

Best tool of the trade for all the above is **HUDSON**



Continuous Integration

Martin Fowler in his landmark article described Continuous Integration as

“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.”



Ten commandments of continuous integration

- *Maintain a Single Source Repository.*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*
- *Automate Deployment*

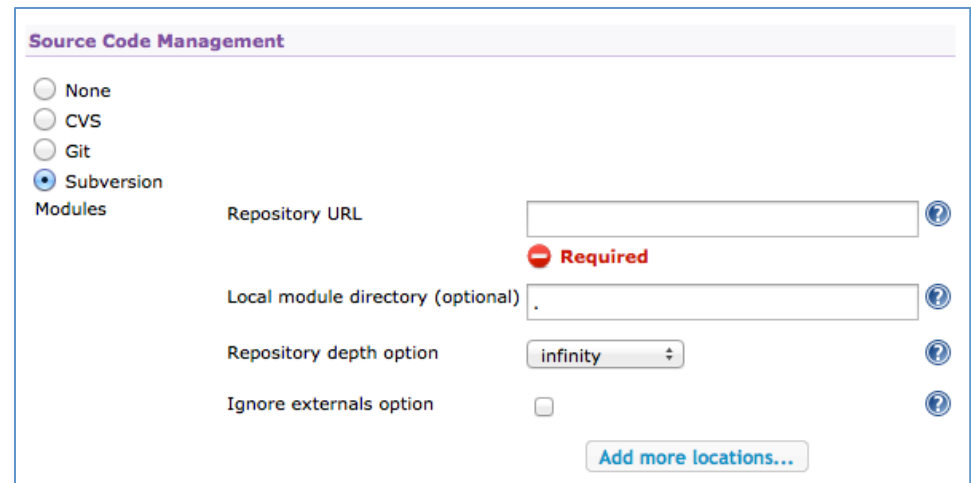
Rest of the slides categorize Hudson Plugins based on these guidelines

Maintain a Single SCM

This principle encourages the project team to use SCM to maintain their source code. Hudson supports various **SCMs** via plugins.

99% of Hudson users use one of

- Git
- CVS
- SVN
- Perforce
- Clearcase
- Mercurial



The screenshot shows the 'Source Code Management' configuration page in Hudson. It features a list of SCM options: None, CVS, Git, and Subversion (selected). Below this, there are fields for 'Repository URL' (marked as required), 'Local module directory (optional)', 'Repository depth option' (set to 'infinity'), and 'Ignore externals option' (unchecked). A button labeled 'Add more locations...' is at the bottom right.

Source Code Management	
<input type="radio"/> None	
<input type="radio"/> CVS	
<input type="radio"/> Git	
<input checked="" type="radio"/> Subversion	
Modules	
Repository URL	<input type="text"/> ?
	Required
Local module directory (optional)	<input type="text"/> ?
Repository depth option	<input type="text" value="infinity"/> ?
Ignore externals option	<input type="checkbox"/> ?
Add more locations...	

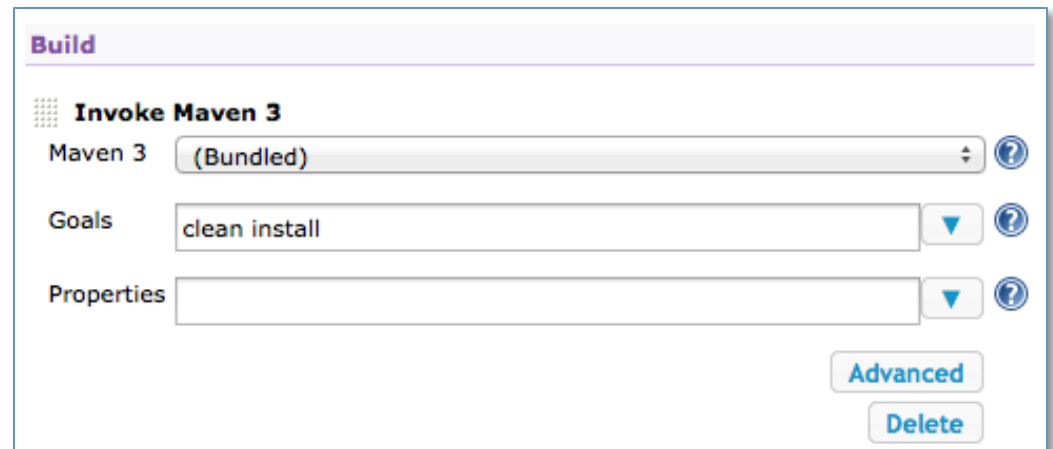
Hudson supports ~20 additional SCM which are used by less than 1% of the users

Automate the Build

Automating the build using a single command is an important principle of a CI build. Hudson supports various **Build Tools** via Plugins.

99% of Hudson users use one of

- Ant
- maven
- gradle
- MSBuild
- Nant
- Rake



The screenshot shows the configuration interface for the 'Invoke Maven 3' plugin in Hudson. The interface is titled 'Build' and includes the following fields and controls:

- Invoke Maven 3**: The plugin name, indicated by a grid icon.
- Maven 3**: A dropdown menu currently set to '(Bundled)', with a help icon (question mark) to its right.
- Goals**: A text input field containing 'clean install', with a dropdown arrow and a help icon to its right.
- Properties**: An empty text input field, with a dropdown arrow and a help icon to its right.
- Advanced**: A button to toggle advanced options.
- Delete**: A button to remove the configuration.

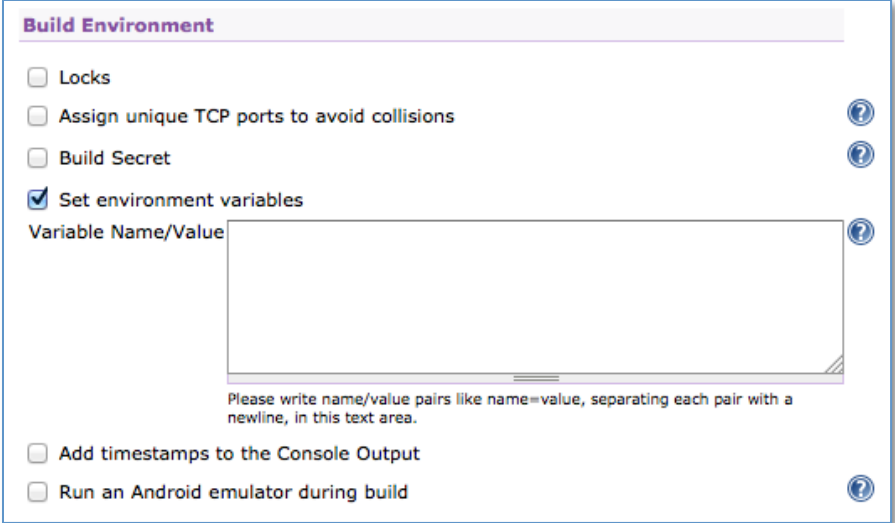
Hudson supports ~40 additional build tools which are used by less than 1% of the users

Automate the Build (Cont..)

Before the build starts, the build must be prepared for proper build. Hudson must support various type **Build Wrappers**. They can be used to post process the build results. Hudson supports various Build wrappers via Plugins. Build wrappers are usually very specific to the Software Project.

Few useful plugins used by 99% of the users are

- Lock & Latch
- Setenv
- EnvInject



The screenshot shows the 'Build Environment' configuration page in Hudson. It features a list of checkboxes for various options: 'Locks', 'Assign unique TCP ports to avoid collisions', 'Build Secret', 'Set environment variables' (which is checked), 'Add timestamps to the Console Output', and 'Run an Android emulator during build'. Below the 'Set environment variables' option is a text area labeled 'Variable Name/Value' with a help icon. A note below the text area reads: 'Please write name/value pairs like name=value, separating each pair with a newline, in this text area.' There are also help icons next to the 'Assign unique TCP ports to avoid collisions', 'Build Secret', and 'Run an Android emulator during build' options.

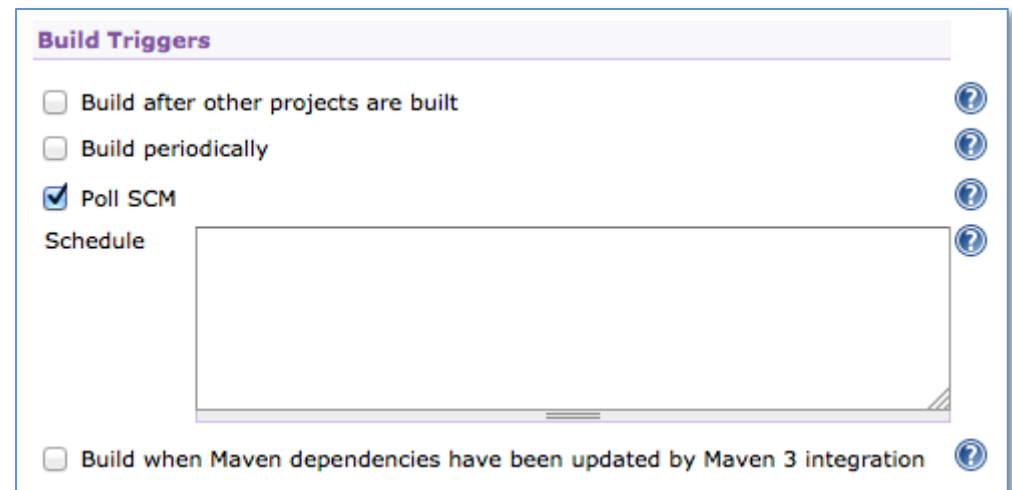
Hudson supports ~25 additional build wrappers which are used by less than 1% of the users

Every commit should build the mainline on an integration machine

Automating the build based on user commit is part of CI. Hudson supports various **Build Triggers** via plugins.

99% of Hudson users use one of

- SCM trigger
- Upstream/Downstream
- Gerrit Trigger
- URL Change
- Scheduled
- Parameterized Trigger



The screenshot shows the 'Build Triggers' configuration page in Hudson. It features a title bar 'Build Triggers' and a list of checkboxes for different trigger types. The 'Poll SCM' option is checked, while others are unchecked. Below the checkboxes is a 'Schedule' text area. At the bottom, there is a checkbox for 'Build when Maven dependencies have been updated by Maven 3 integration'. Each checkbox has a help icon (question mark) to its right.

Trigger Type	Checked
Build after other projects are built	<input type="checkbox"/>
Build periodically	<input type="checkbox"/>
Poll SCM	<input checked="" type="checkbox"/>
Schedule	<input type="text"/>
Build when Maven dependencies have been updated by Maven 3 integration	<input type="checkbox"/>

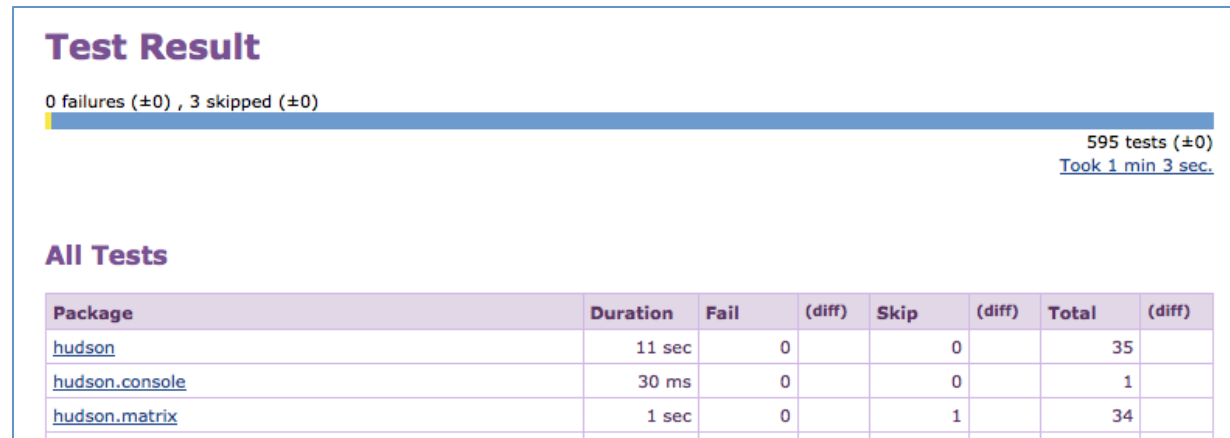
Hudson supports ~15 additional build triggers which are used by less than 1% of the users

Make your build self-testing

CI build is not about catching compilation error but also catching bugs more quickly and efficiently. Hudson supports various **Unit Testing Frameworks** via Plugins.

99% of Hudson users use one of

- jUnit
- nUnit
- Selenium
- CppUnit
- TestNg
- xUnit



Test Result

0 failures (±0) , 3 skipped (±0)

595 tests (±0)
Took 1 min 3 sec.

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
hudson	11 sec	0		0		35	
hudson.console	30 ms	0		0		1	
hudson.matrix	1 sec	0		1		34	

Hudson supports ~10 additional Unit Test which are used by less than 1% of the users

Make your build self-testing (Code Coverage)

Self testing is best achieved if there is uniform code coverage. Hudson supports various **Code Coverage Tools** via Plugins




99% of Hudson users use one of




- Clover
- Cobertura
- Emma
- Serenity
- Sonar
- NCover

Publish Clover Coverage Report

Clover report directory

Specify the path to the directory that contains the clover.xml report file, relative to [the workspace root](#). Clover must be configured to generate XML reports for this plugin to function fully.

Coverage Metric Targets	% Methods	% Conditionals	% Statements
	<input type="text" value="70"/>	<input type="text" value="80"/>	<input type="text" value="80"/>
	<input type="text" value="40"/>	<input type="text" value="60"/>	<input type="text" value="50"/>
	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>

Configure health reporting thresholds.
For the  row, leave blank to use the default values (i.e. 70, 80, and 80 for methods, conditionals and statements respectively).
For the  and  rows, leave blank to use the default values (i.e. 0).

Hudson supports ~2 additional Code Coverage which are used by less than 1% of the users

Make your build self-testing (Code Analysis)

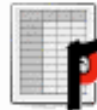
Static Analysis improves the confidence of Self testing. Hudson supports various **Static Code Analysis Tools** via Plugins

99% of Hudson users use one of

- Checkstyle
- PMD
- Dry
- Findbugs
- Crap4J
- Warnings
- CCM
- Violations



FindBugs: [1 warning](#) in 14 FindBugs files.



PMD: [719 warnings](#) in 3 PMD files.



Checkstyle: [6 warnings](#) in 19 Checkstyle files.

- ◆ [6 new warnings](#)

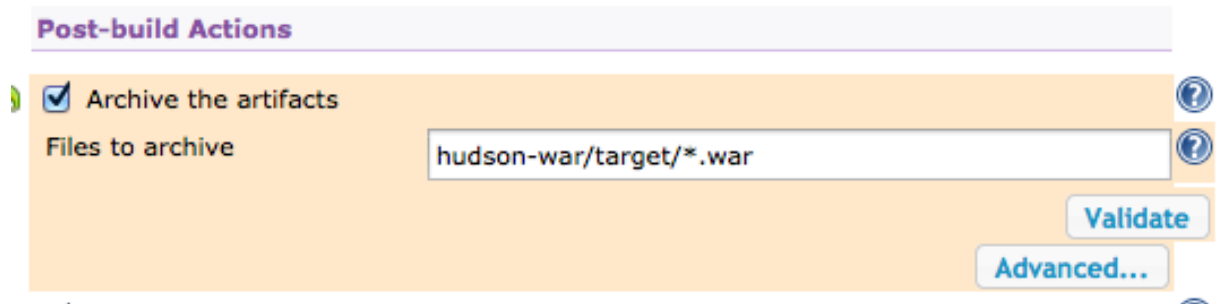
Hudson supports ~2 additional Code Analysis which are used by less than 1% of the users

Make it easy for everyone to get latest executable

Make the build artifacts to stakeholders is important in CI. Hudson supports various **Artifact Uploaders** via Plugins.

95% of Hudson users use one of

- CopyArtifacts
- SCP
- FTP Publisher
- Artifactory
- Maven Release
- HTML Publisher



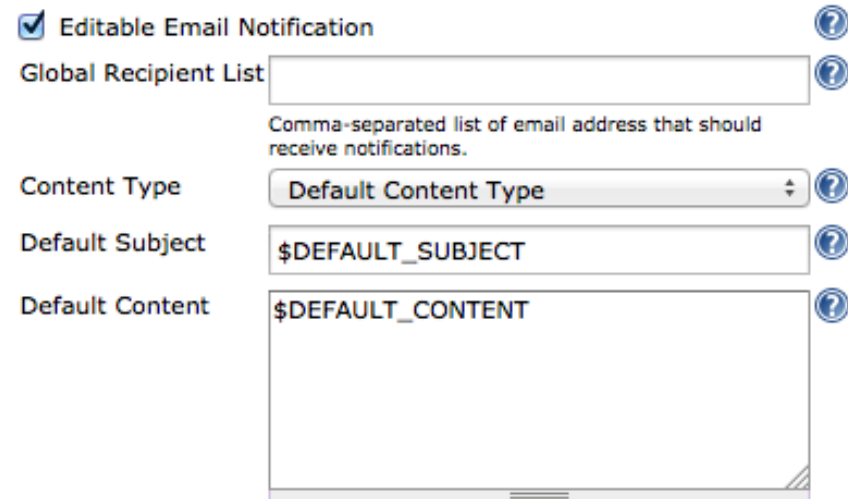
Hudson supports ~10 additional Artifact Uploaders used by less than 5% of the users

Everyone can see what is happening

Communicate the state of the build especially if it is broken . Hudson supports various **Build Notifiers** via plugins.

95% of Hudson users use one of

- Email
- Email-Ext
- IRC
- Jabber
- SMS



The screenshot shows the configuration for an Editable Email Notification. It includes a checked checkbox for 'Editable Email Notification', a text input for 'Global Recipient List' with a help icon, a dropdown menu for 'Content Type' set to 'Default Content Type', a text input for 'Default Subject' containing '\$DEFAULT_SUBJECT', and a text area for 'Default Content' containing '\$DEFAULT_CONTENT'. Each input field has a help icon to its right.

Editable Email Notification ?

Global Recipient List ?

Comma-separated list of email address that should receive notifications.

Content Type ?

Default Subject ?

Default Content ?

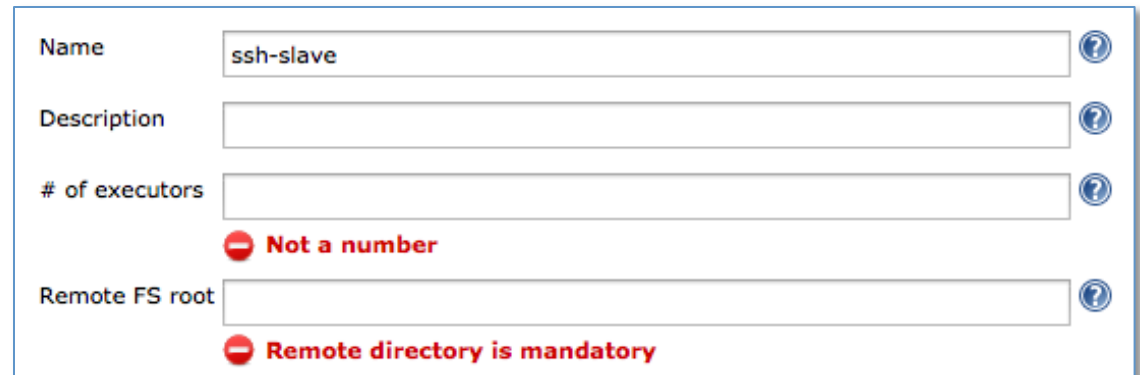
Hudson supports ~20 additional Build Notifiers used by less than 5% of the users

Test in a clone of the production environment

The build must happen in various slaves that clone the production environment. Hudson supports various kind of **Slave Management** via plugins so builds can happen in clones of production environment.

99% of Hudson users use one of

- SSH Slave
- Windows Slave
- Slave Status
- EC2
- Virtual Box
- JCloud



The image shows a screenshot of the Hudson Slave Management configuration form. The form has four input fields: 'Name' (containing 'ssh-slave'), 'Description', '# of executors', and 'Remote FS root'. Each field has a help icon (question mark) to its right. Below the '# of executors' field, there is a red error message: 'Not a number'. Below the 'Remote FS root' field, there is a red error message: 'Remote directory is mandatory'.

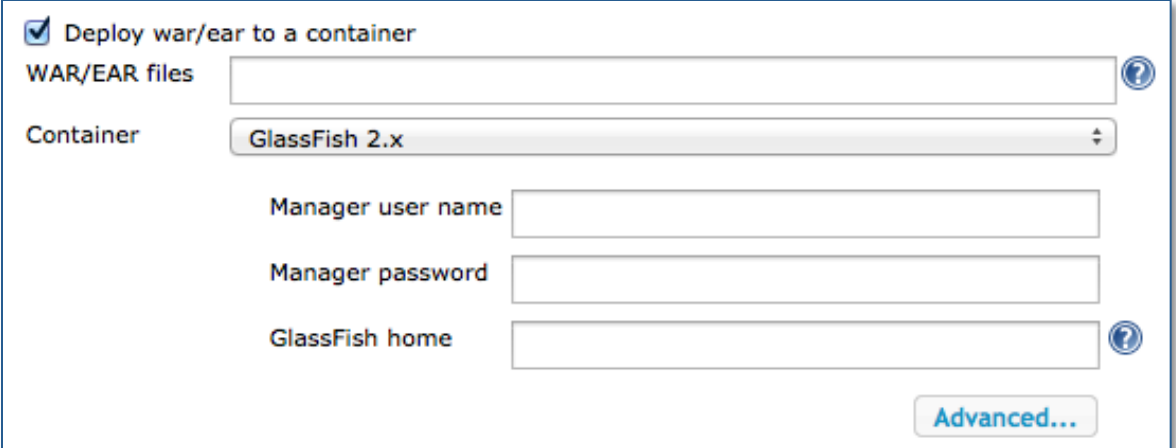
Hudson supports ~15 additional Slave Management Tools used by less than 1% of the users

Automate deployment

One of the CI best practices is to automate the deployment. Hudson supports various type of **Deployment or External Tool Integration** via plugins.

95% of Hudson users use one of

- Tomcat
- Websphere
- Weblogic
- Jboss
- JRebel
- IIS



The screenshot shows a configuration panel for deploying WAR/EAR files to a container. The 'Deploy war/ear to a container' checkbox is checked. The 'WAR/EAR files' field is empty. The 'Container' dropdown is set to 'GlassFish 2.x'. Below this, there are three input fields: 'Manager user name', 'Manager password', and 'GlassFish home', all of which are currently empty. There are help icons (question marks) next to the 'WAR/EAR files' and 'GlassFish home' fields. An 'Advanced...' button is located at the bottom right of the panel.

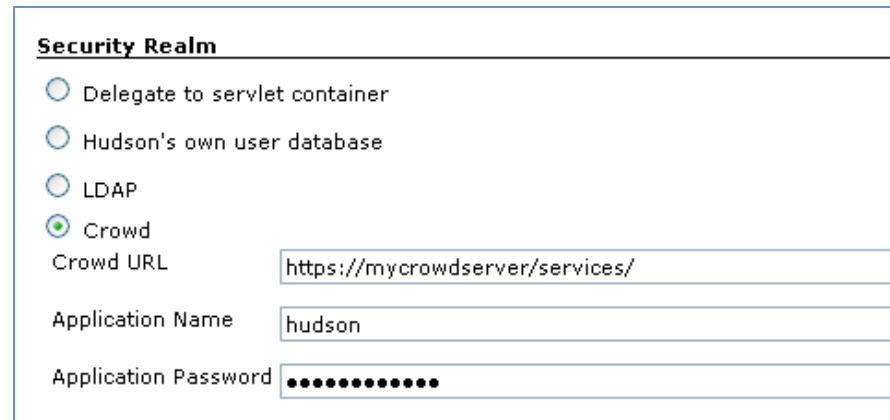
Hudson supports ~40 additional External Tool Integration used by less than 5% of the users

Hudson Security

Hudson supports various type of **Authentication & Authorization** mechanisms via plugins.

99% of Hudson users use one of

- LDAP
- Active Directory
- Crowd
- OpenId
- PwAuth (Unix)



The screenshot shows the 'Security Realm' configuration page in Hudson. It features a list of radio buttons for selecting an authentication method: 'Delegate to servlet container', 'Hudson's own user database', 'LDAP', and 'Crowd'. The 'Crowd' option is selected. Below the radio buttons are three input fields: 'Crowd URL' with the value 'https://mycrowdserver/services/', 'Application Name' with the value 'hudson', and 'Application Password' which is masked with dots.

Security Realm	
<input type="radio"/>	Delegate to servlet container
<input type="radio"/>	Hudson's own user database
<input type="radio"/>	LDAP
<input checked="" type="radio"/>	Crowd
Crowd URL	<input type="text" value="https://mycrowdserver/services/"/>
Application Name	<input type="text" value="hudson"/>
Application Password	<input type="password" value="....."/>

Hudson supports ~15 additional Authentication used by less than 5% of the users

Hudson UI Configuration

Various parts of Hudson UI can be configured or Augmented. These **UI configurations** are supported via Plugins.

95% of Hudson users use one of

- Chuck Norris
- Disk Usage
- Plot
- Build Dependency
- Radiator View
- Xfpanel
- Green Balls
- Build Pipeline
- Nested View
- Downstream Build View
- Dashboard View



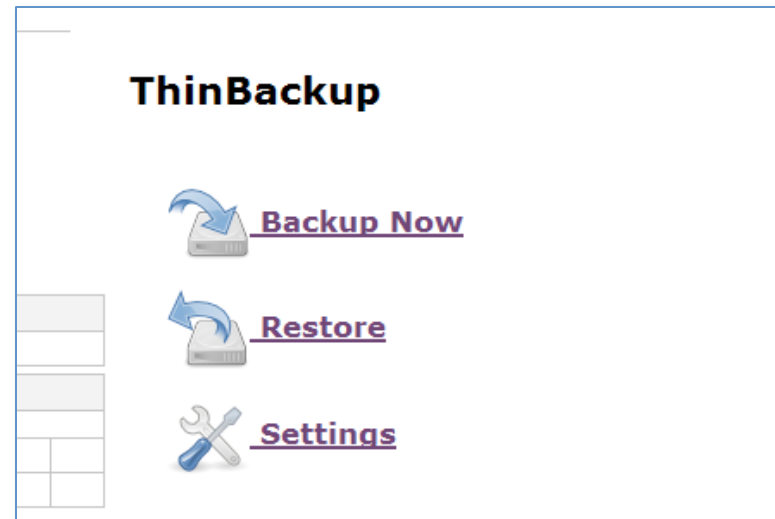
Hudson supports ~50 additional UI configurations used by less than 5% of the users

Hudson Utilities

Various **Utilities**, supported via Plugins can be added to Hudson to improve day to day CI activities.

95% of Hudson users use one of

- Translation
- Build timeout
- backup
- Log parser
- Audit Trail
- Status Monitor
- Global Build Stats
- Project Stats
- Slave Status



Hudson supports ~20 additional Utilities used by less than 5% of the users