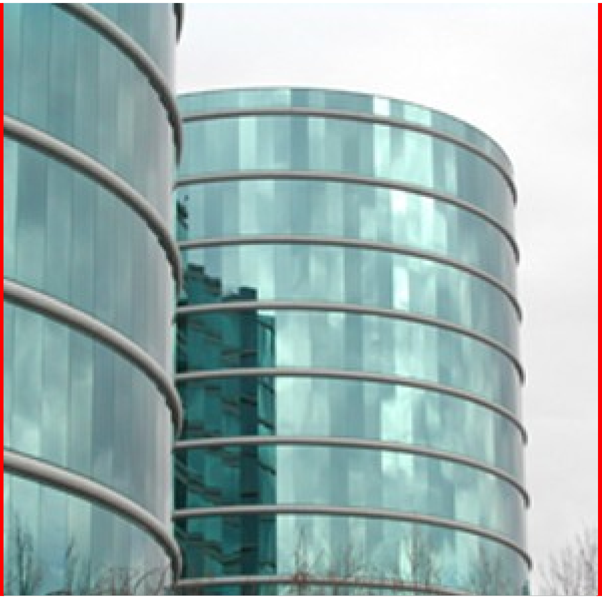


Introducing Hudson

Winston Prakash

What is Hudson?

Hudson is an open source “continuous integration” (CI) server. A CI server can do various tasks like

- check-out source code
- build and test the project
- publish the results
- communicate the results to team members

and much more ..

Key Features of Hudson

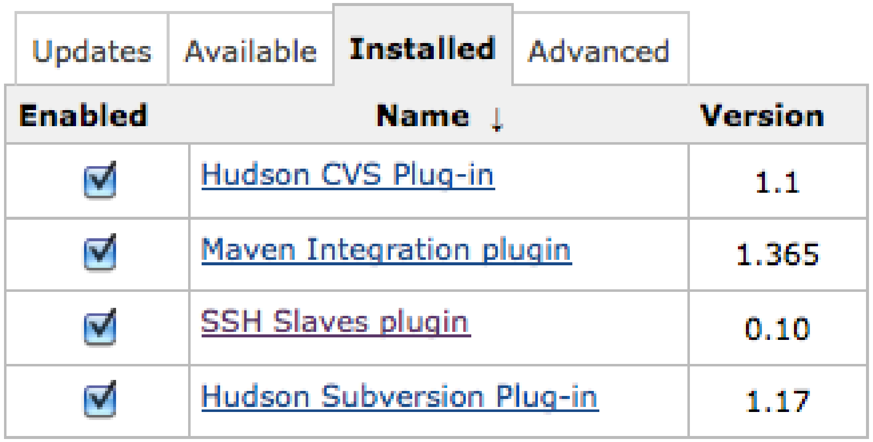
- Easy installation
- Easy configuration
- Web based interface
- Distributed Builds
- Unit test Reporting
- File Fingerprinting
- Build status Notification
- Extendable with plugins

Plugins installed by default

Hudson is a extendable execution platform. Functionalities are extended via plugins.

By default only 4 plugins are installed, which supports

- CVS
- SVN
- Maven
- SSH



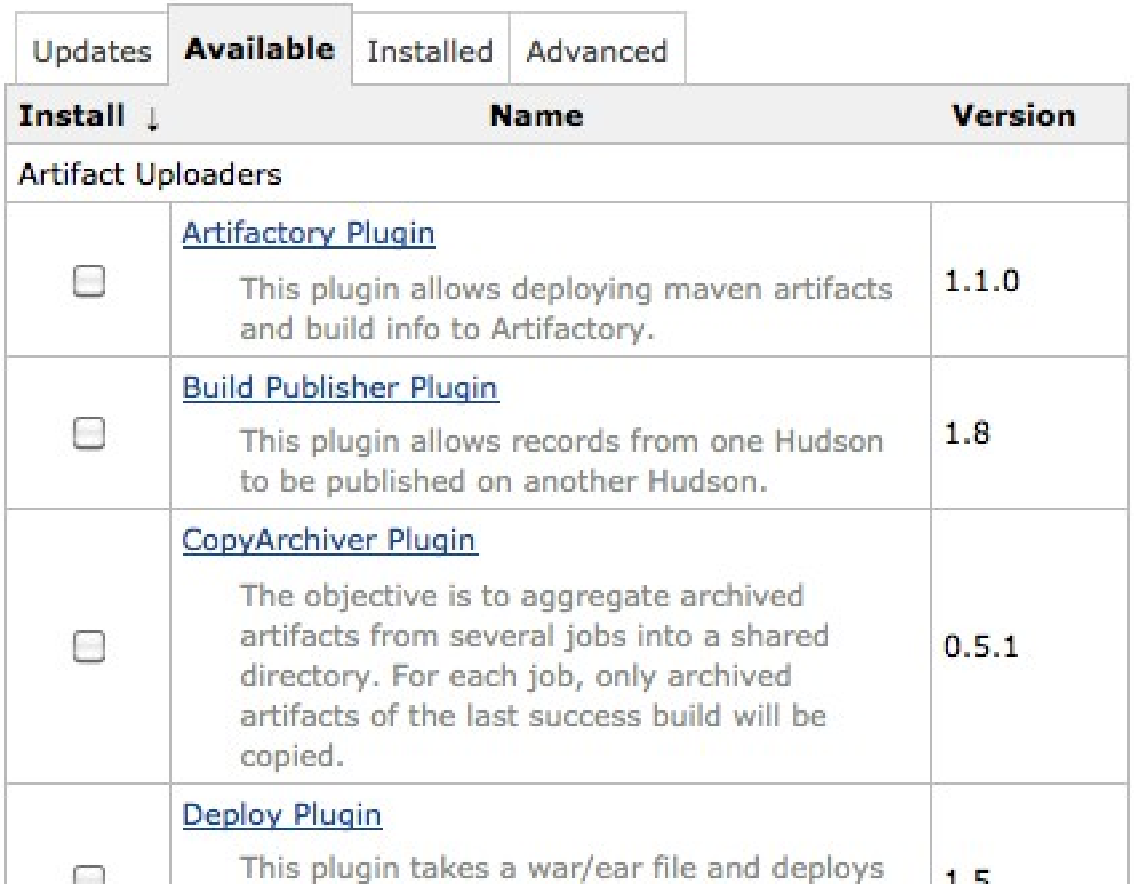
Enabled	Name ↓	Version
☑	Hudson CVS Plug-in	1.1
☑	Maven Integration plugin	1.365
☑	SSH Slaves plugin	0.10
☑	Hudson Subversion Plug-in	1.17

Available Plugins

However, there are more than 150 plugins available for various topics such as

- Artifact Uploaders
- Authentication
- Build Notifiers
- Build Reports
- Build Tools
- Build Triggers
- Build Wrappers
- Cluster Management

etc..

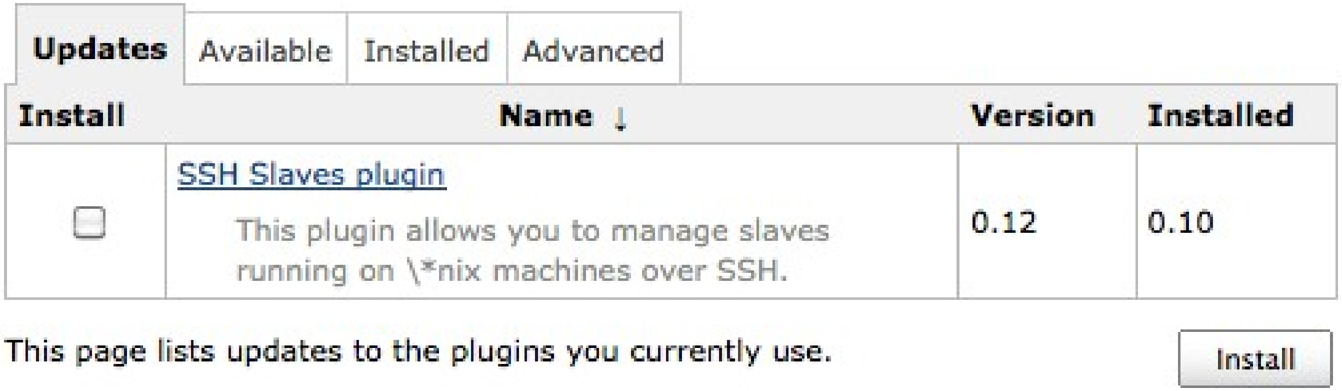


Updates | **Available** | Installed | Advanced

Install ↓	Name	Version
Artifact Uploaders		
☐	Artifactory Plugin This plugin allows deploying maven artifacts and build info to Artifactory.	1.1.0
☐	Build Publisher Plugin This plugin allows records from one Hudson to be published on another Hudson.	1.8
☐	CopyArchiver Plugin The objective is to aggregate archived artifacts from several jobs into a shared directory. For each job, only archived artifacts of the last success build will be copied.	0.5.1
☐	Deploy Plugin This plugin takes a war/ear file and deploys	1.5

Plugin Updates

When an update to an installed plugin is released, it is available to the user via the update tab in the plugin page (Home → Manage Hudson → Manage Plugins)



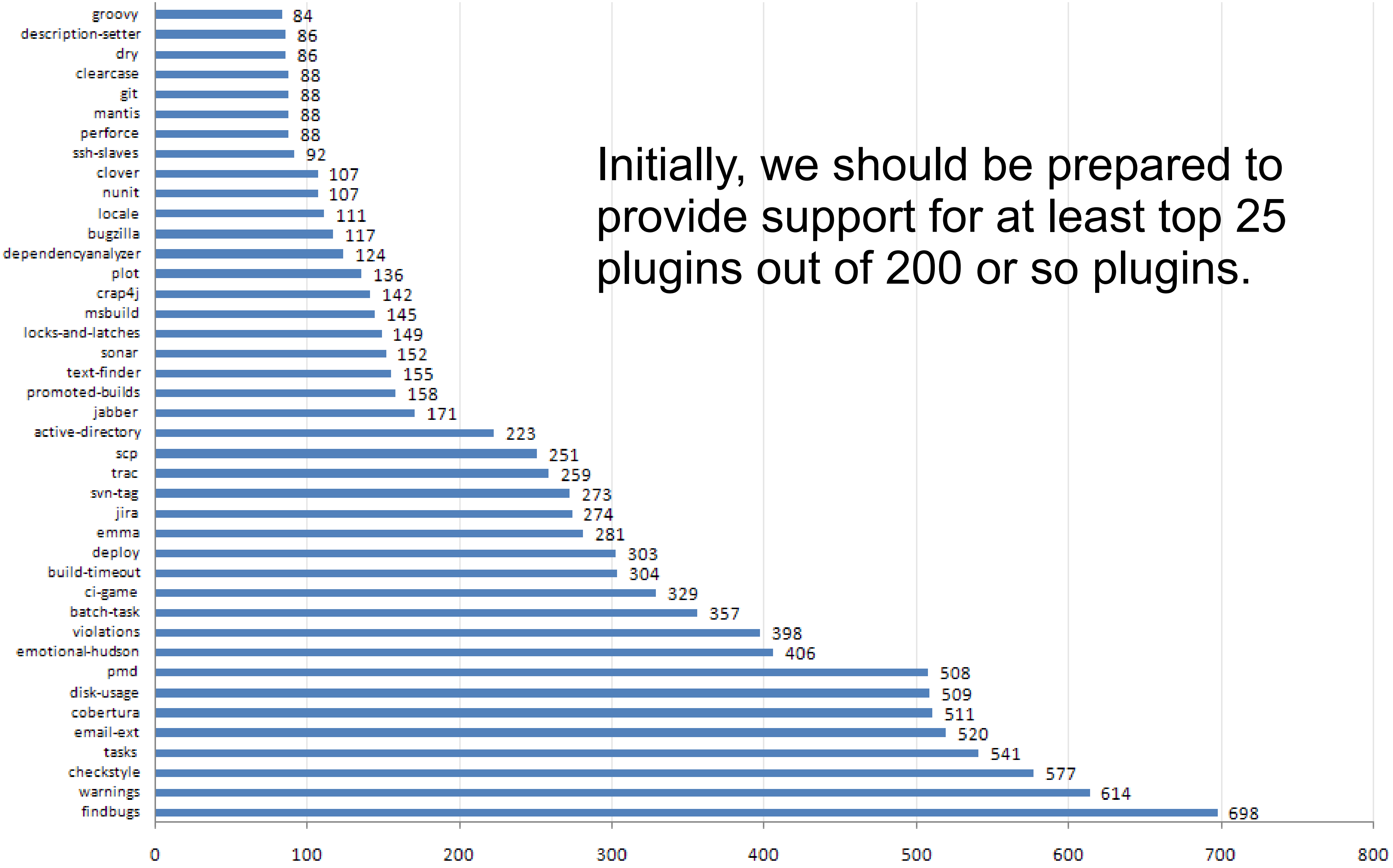
Install	Name ↓	Version	Installed
☐	SSH Slaves plugin This plugin allows you to manage slaves running on *nix machines over SSH.	0.12	0.10

This page lists updates to the plugins you currently use.

Trouble Shooting:

- *If a newly installed plugin doesn't show up or updated plugin is behaving erratically, shutdown and restart Hudson.*
- *For failed connections while using Hudson from behind a FireWall, a proxy configuration can be set via Hudson Configuration page.*

Popular Plugins
(based on 2009 survey)



Initially, we should be prepared to provide support for at least top 25 plugins out of 200 or so plugins.

Installing and Running Hudson

There is no Installation step!

Hudson can be run in two modes

•Standalone mode
Invoke using JNLP - *https://hudson.dev.java.net/hudson.jnlp*
From command line - *java -jar hudson.war*

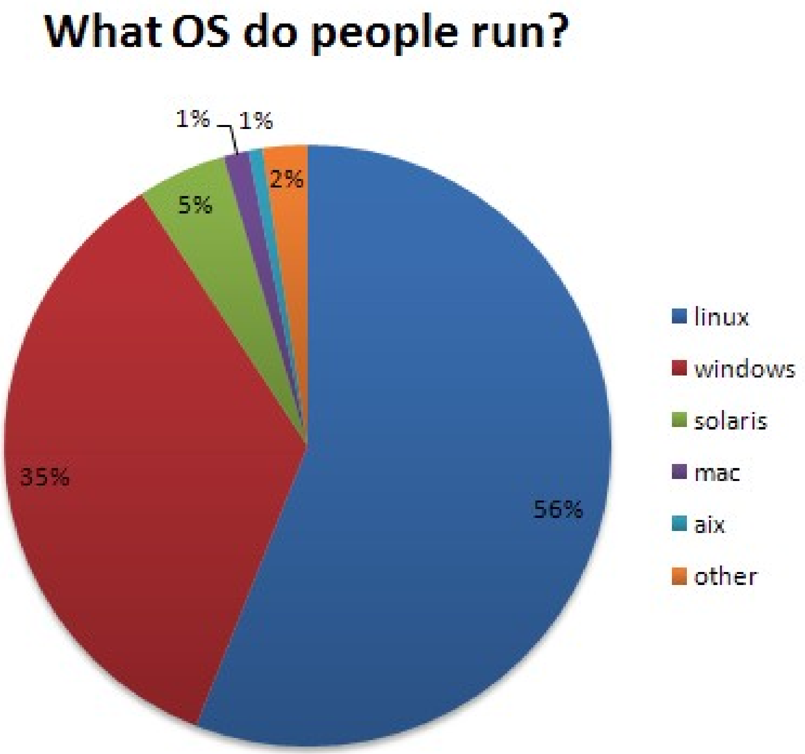
•Deploy hudson.war to any one of the following JavaEE containers

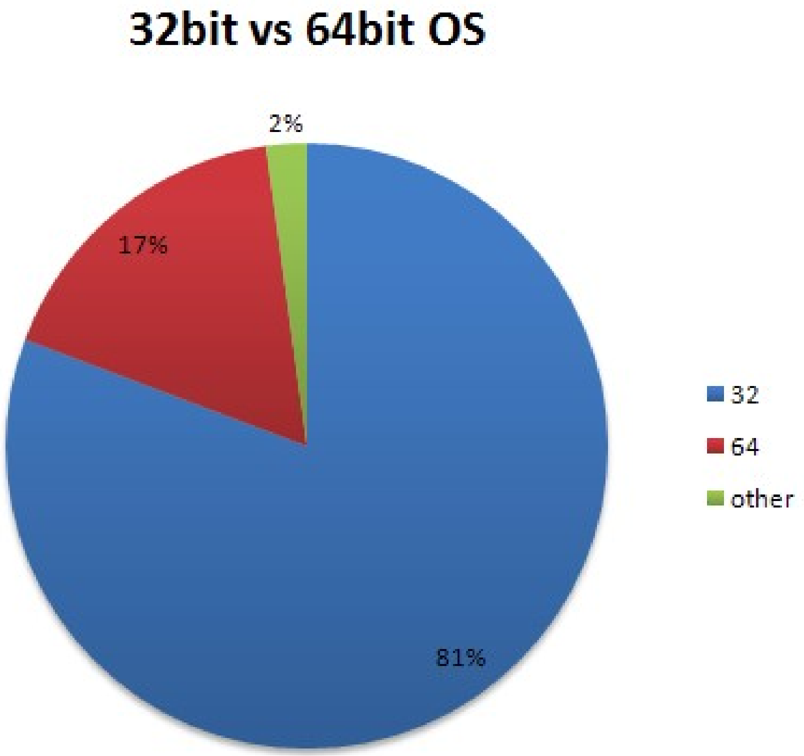
Glassfish
Websphere
Jboss
Jetty
Tomcat
Winstone

Note: While running in standalone mode, a builtin JavaEE container (**winstone**) is used.

Operating System used to run Hudson

(based on 2009 survey)

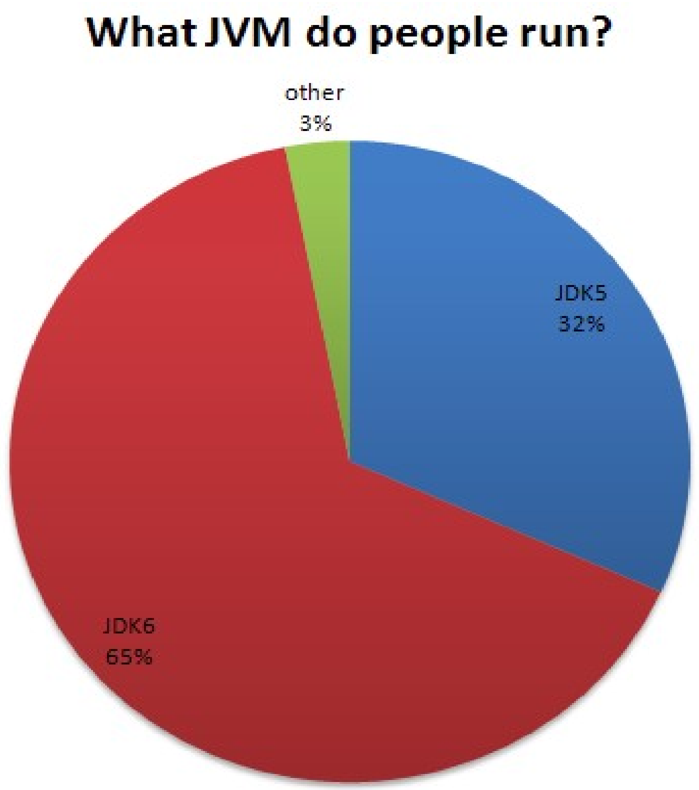

What OS do people run?


32bit vs 64bit OS

Linux and windows seems to be the primary OS and need good support for Hudson installations on both of them.

What JDK is used to run Hudosn

(based on 2009 survey)



As of Jan 2009 JDK 5 is still used by 32%. But after 18 months more users must have switched to JDK 6. But we still be prepared to support JDK 5 too.

Accessing Hudson

While running in standalone mode, the default port used is 8080

Access Hudson using the URL

http://<Hostname>:<port> (http://localhost:8080)

While deployed in a container, Hudson is accessed via the context root “hudson”. If Hudson is deployed to a JavaEE server on a machine (jag2.foundary.suncom on the port 80), then it is accessed via the URL

Ex. http://jag2.foundry.sun.com/hudson/

Note:

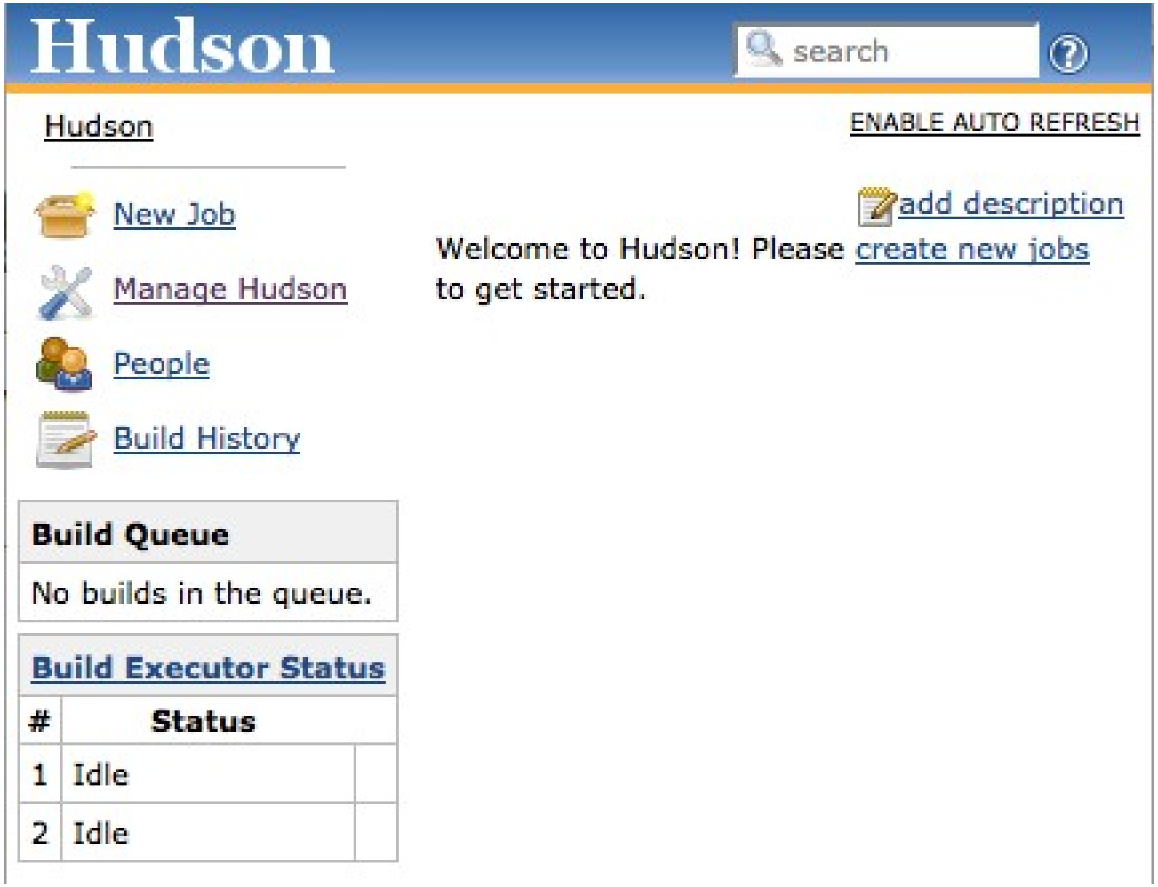
To change the default port while running Hudson via command line use

```
java -jar hudson.war –httpPort 8787
```

Accessing Hudson Continued..

By default hudson starts with out any security setup.

Next step is to Manage Hudson to configure the security realm, add more plugins etc



Welcome to Hudson! Please create new jobs to get started.

Build Queue

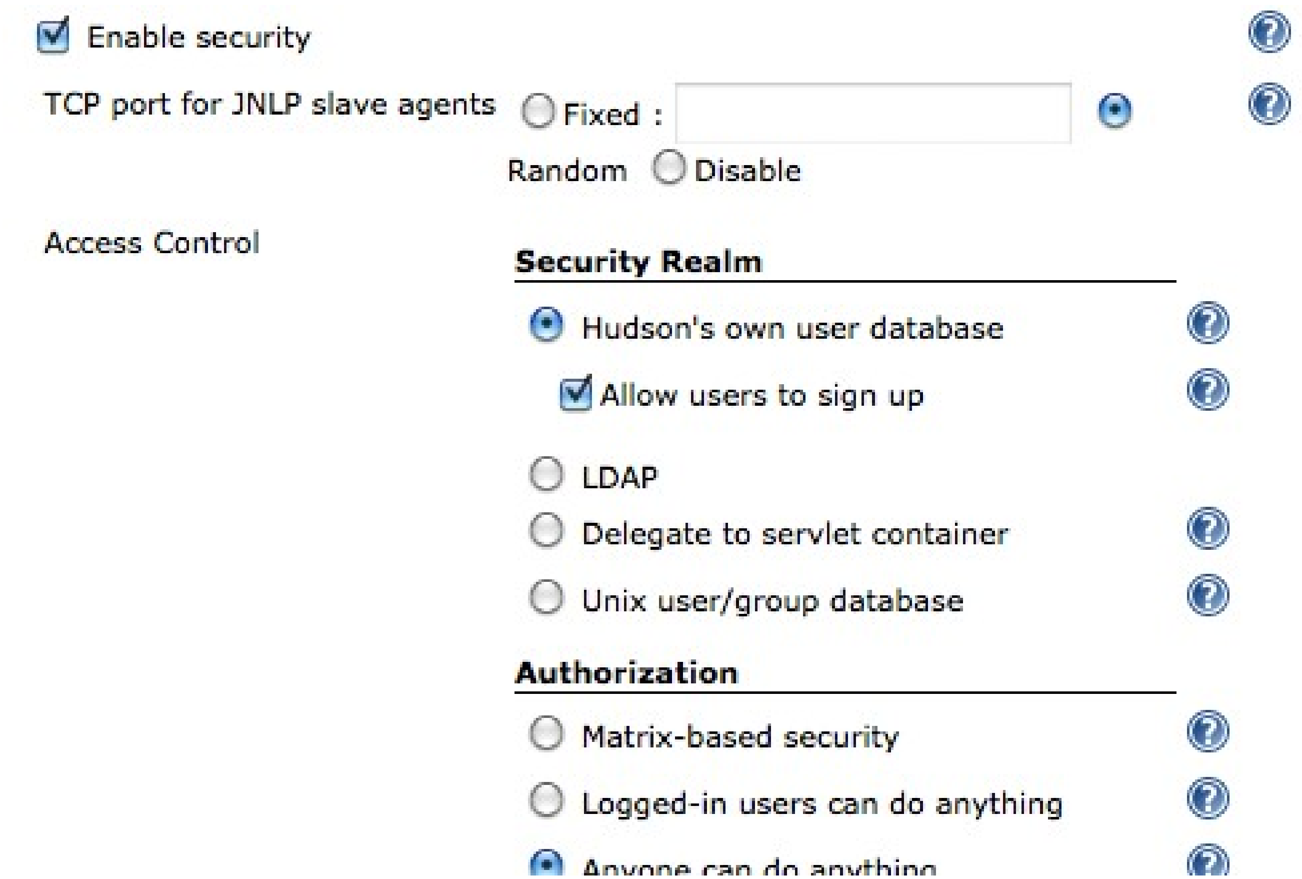
No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Configuring Hudson Security

Hudson security is enabled via Home → Manage Hudson → Configure → Enable Security



☑ Enable security

TCP port for JNLP slave agents ○ Fixed : ● Random ○ Disable

Access Control

Security Realm

● Hudson's own user database

☑ Allow users to sign up

○ LDAP

○ Delegate to servlet container

○ Unix user/group database

Authorization

○ Matrix-based security

○ Logged-in users can do anything

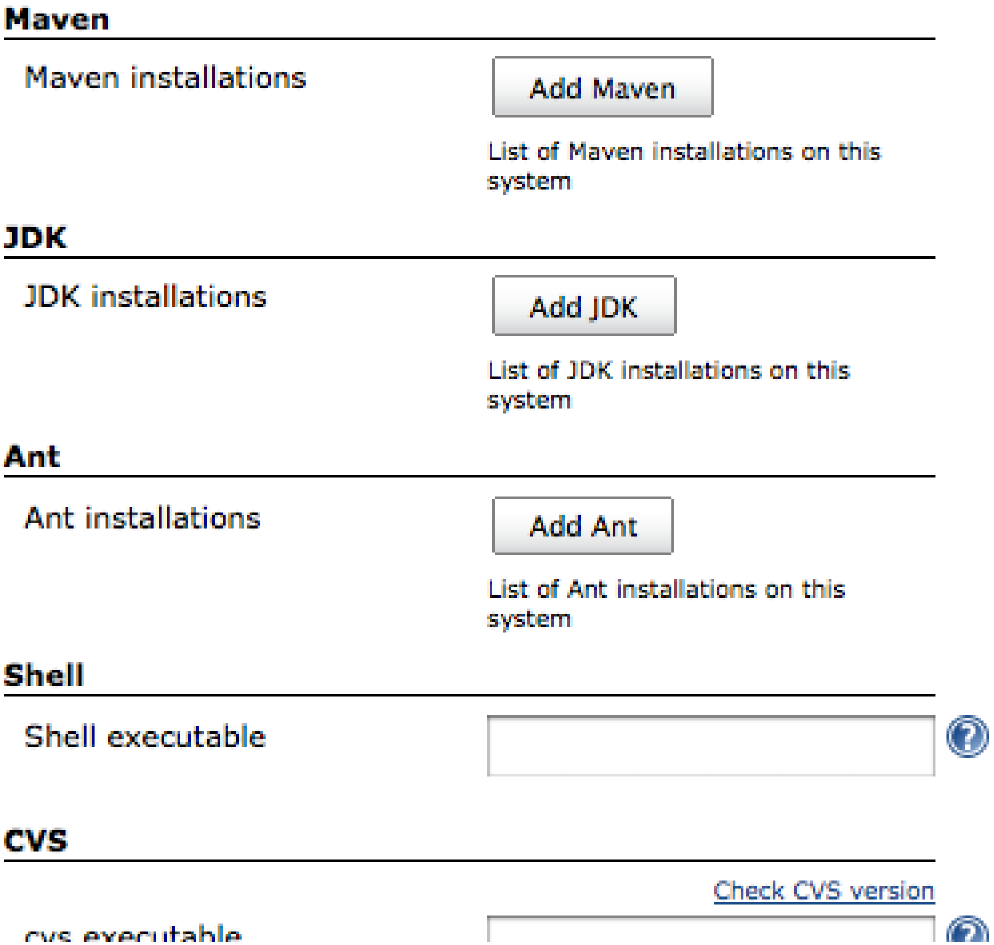
● Anyone can do anything

While using builtin user database, if users are not allowed to sign up, then an admin user is first created.

Configuring Build related artifacts

Next step is to tell Hudson where to find the installations of JDK, CVS, SVN, Maven etc.

On a Unix or Linux machine Hudson will figure out from the path setting, but on windows machine, the exec path may need to be specified.



Maven

Maven installations

List of Maven installations on this system

JDK

JDK installations

List of JDK installations on this system

Ant

Ant installations

List of Ant installations on this system

Shell

Shell executable

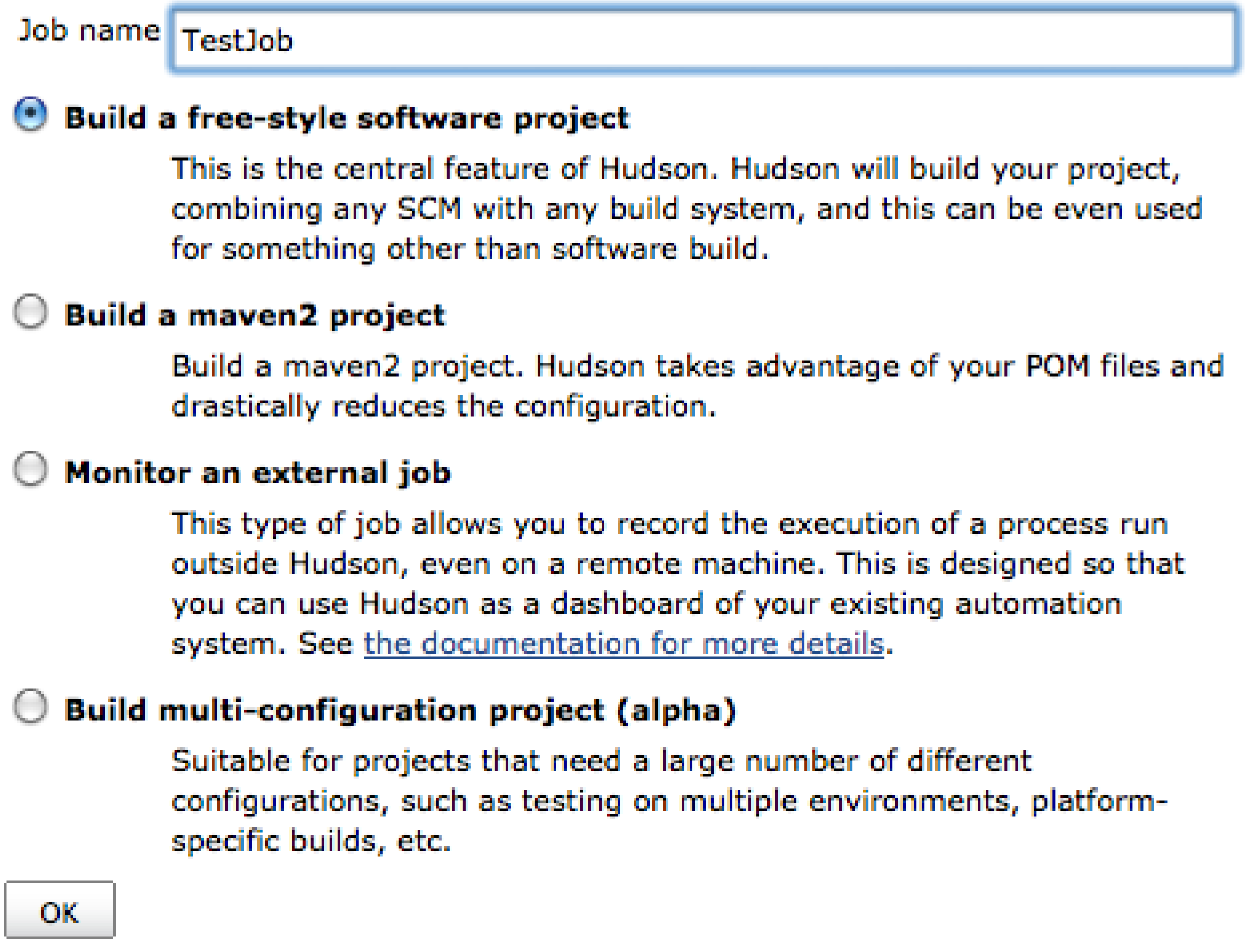
CVS

Check CVS version

cvs executable

Creating Hudson Free-Style Project

Once the Hudson is fully configured, next step is to create a job (Free-Style Software Project) to execute by clicking on the “New Job” link.



Job name: TestJob

● **Build a free-style software project**

This is the central feature of Hudson. Hudson will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

○ **Build a maven2 project**

Build a maven2 project. Hudson takes advantage of your POM files and drastically reduces the configuration.

○ **Monitor an external job**

This type of job allows you to record the execution of a process run outside Hudson, even on a remote machine. This is designed so that you can use Hudson as a dashboard of your existing automation system. See the documentation for more details.

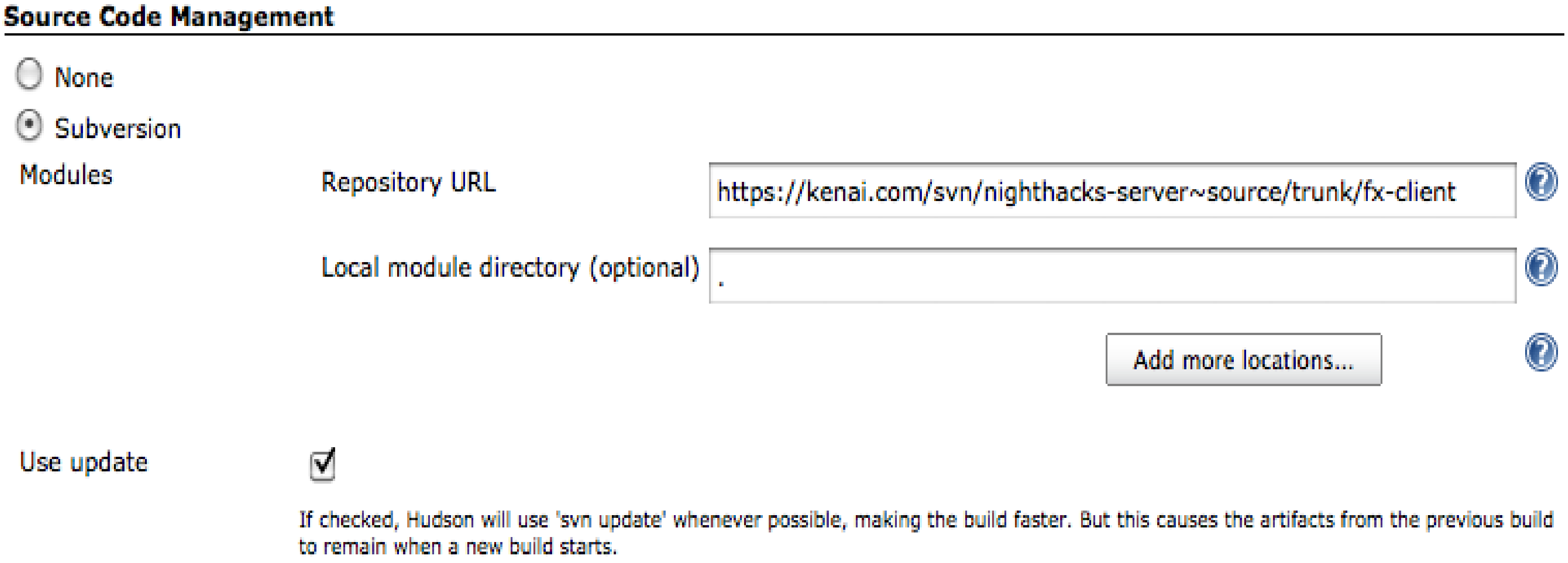
○ **Build multi-configuration project (alpha)**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Configuring the Job

If the job is for building a project sources, next step is to provide a Source Code Management information from where the sources can be downloads.

By default only CVS and Subversion are supported. But plugins are available for other SCM such as Clearcase, Git, perforce, mercurial, VSS, accirev, tfs etc.



Source Code Management

○ None

● Subversion

Modules

Repository URL: https://kenai.com/svn/nighthacks-server~source/trunk/fx-client

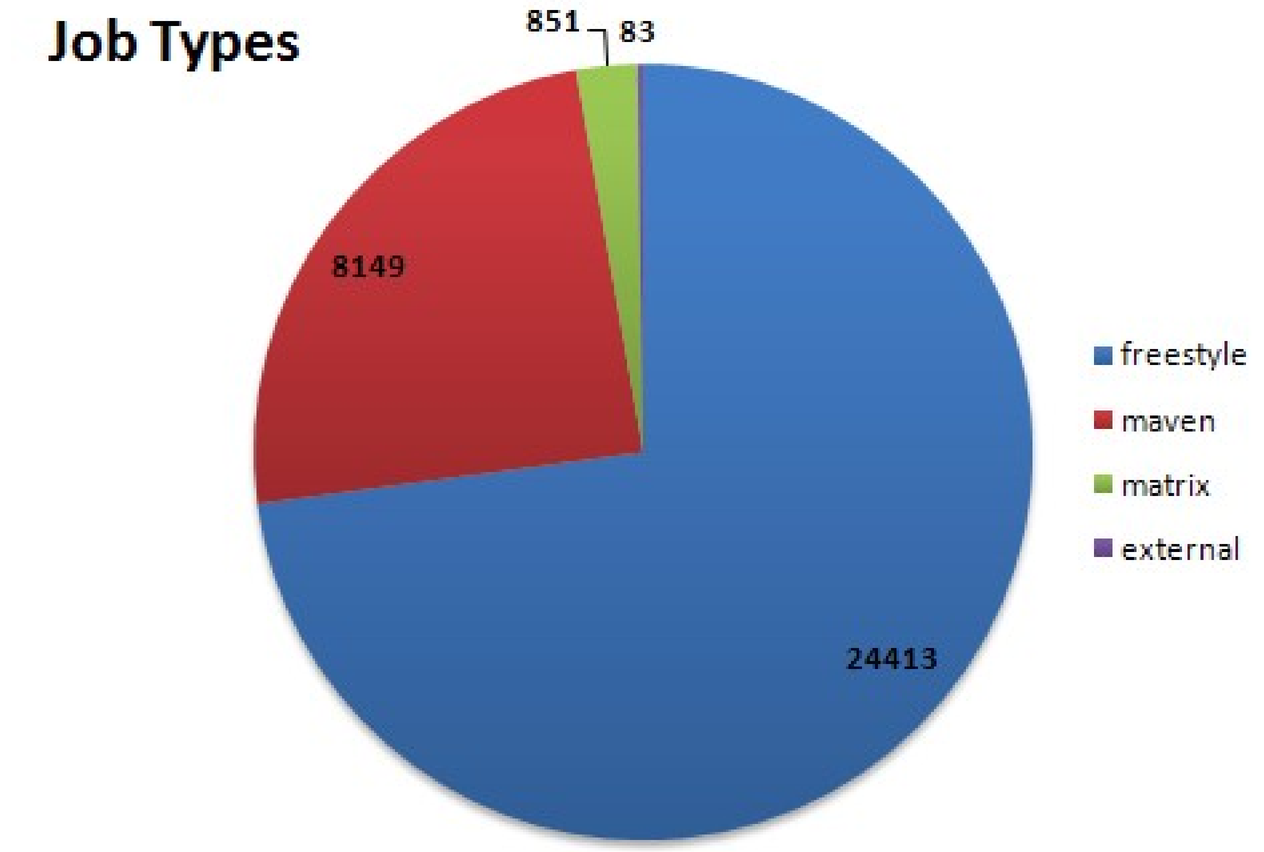
Local module directory (optional): .

Use update ☑

If checked, Hudson will use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Widely used Hudson Job Types

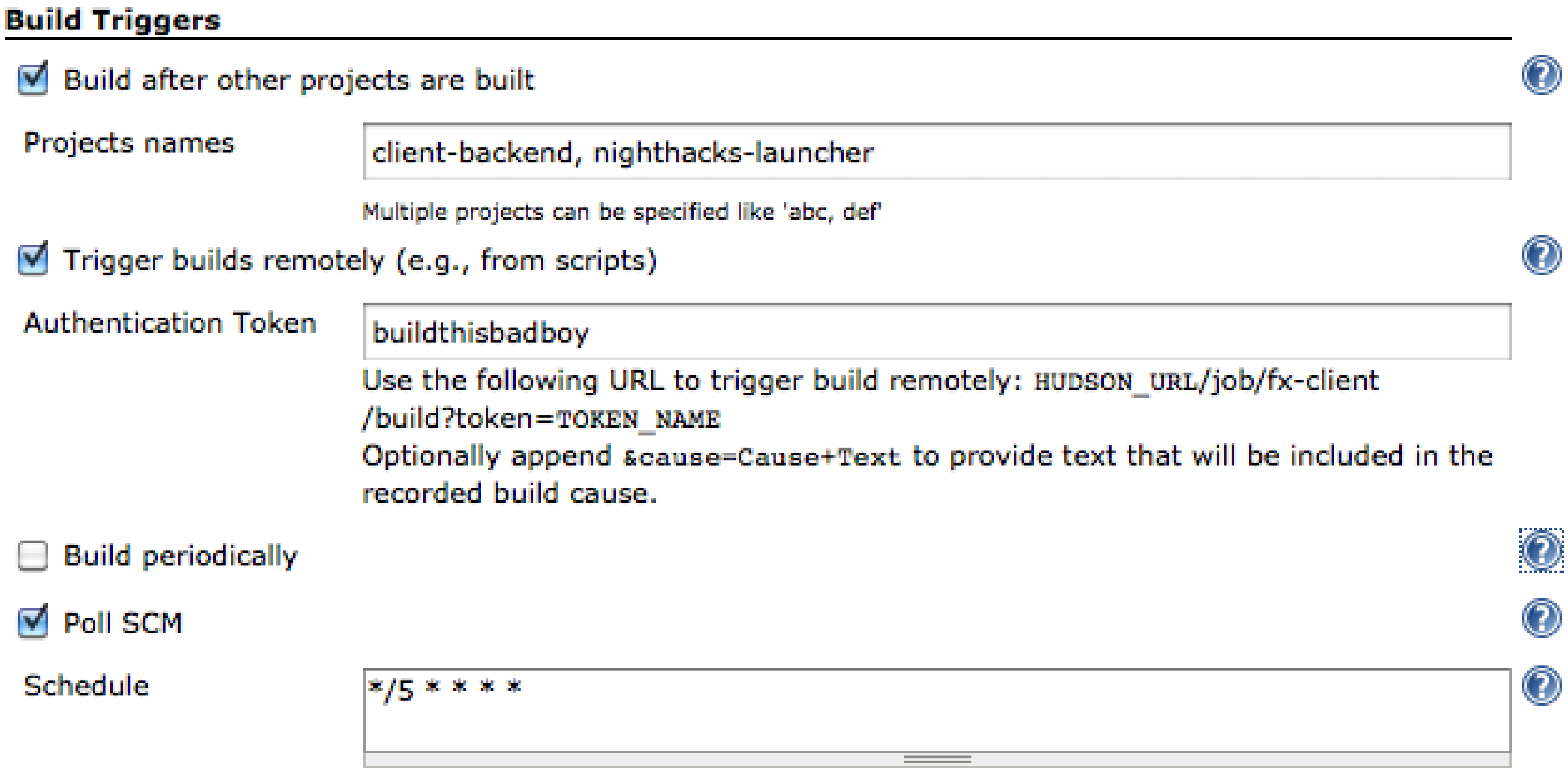
(based on 2009 survey)



Approximately 72% Freestyle 24% Maven

Configuring the Job Continued..

Next we must specify when the build should get triggered. The obvious choice for Software project is when somebody checked into SCM. In the following example, SCM is polled for every 5 minutes to see if any new checkin has happened. Optionally it is possible to make the current project to build after other projects are built.



Build Triggers

☑ Build after other projects are built

Projects names: client-backend, nighthacks-launcher

Multiple projects can be specified like 'abc, def'

☑ Trigger builds remotely (e.g., from scripts)

Authentication Token: buildthisbadboy

Use the following URL to trigger build remotely: HUDSON_URL/job/fx-client/build?token=TOKEN_NAME
Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

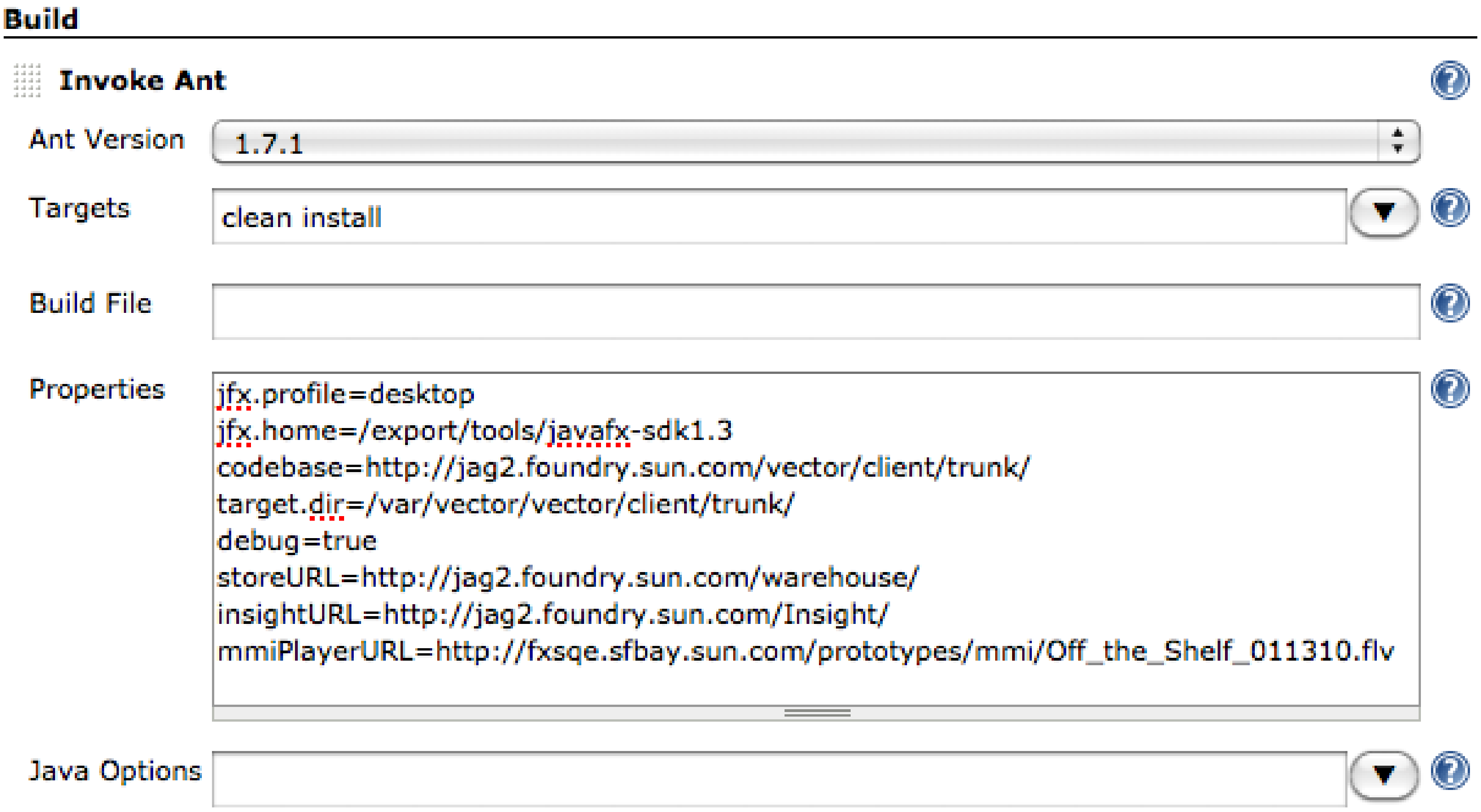
☐ Build periodically

☑ Poll SCM

Schedule: */5 * * * *

Configuring the Job Continued..

The last mandatory step is to tell Hudson how to build. Usually it is done via an ant build. This also provides ways to specify targets, properties and Java Options to ant.



Build

Invoke Ant

Ant Version: 1.7.1

Targets: clean install

Build File:

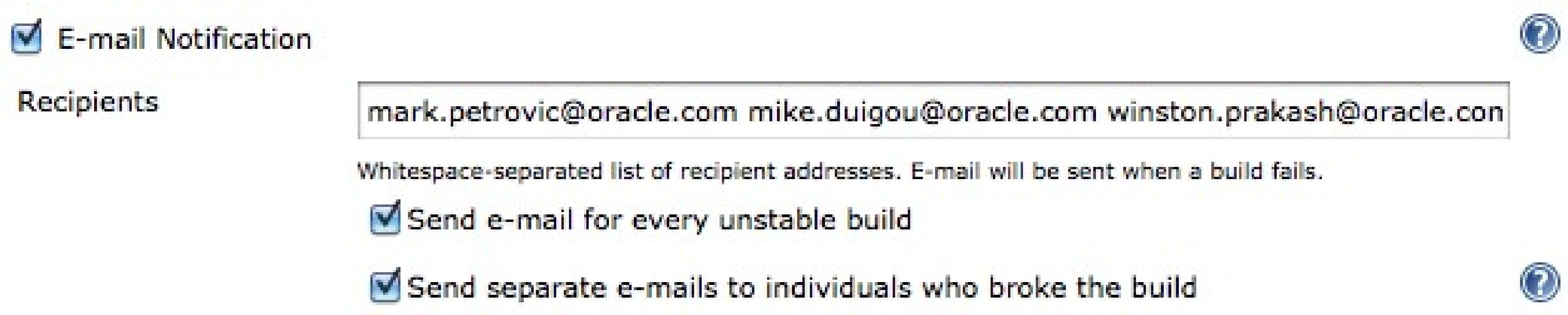
Properties:

```
jfx.profile=desktop
jfx.home=/export/tools/javafx-sdk1.3
codebase=http://jag2.foundry.sun.com/vector/client/trunk/
target.dir=/var/vector/vector/client/trunk/
debug=true
storeURL=http://jag2.foundry.sun.com/warehouse/
insightURL=http://jag2.foundry.sun.com/Insight/
mmiPlayerURL=http://fxsqe.sfbay.sun.com/prototypes/mmi/Off_the_Shelf_011310.flv
```

Java Options:

Configuring the Job Continued..

Another important option is to tell Hudson whom to send e-mail when the builds become unstable.



☑ E-mail Notification

Recipients: mark.petrovic@oracle.com mike.duigou@oracle.com winston.prakash@oracle.con

Whitespace-separated list of recipient addresses. E-mail will be sent when a build fails.

☑ Send e-mail for every unstable build

☑ Send separate e-mails to individuals who broke the build

There are several other options

- Deploy War after build
- Invoke post batch jobs after build completes
- Archive the artifacts associated with build
- Update relevant JIRA Issue
- Plot build data

to name a few.

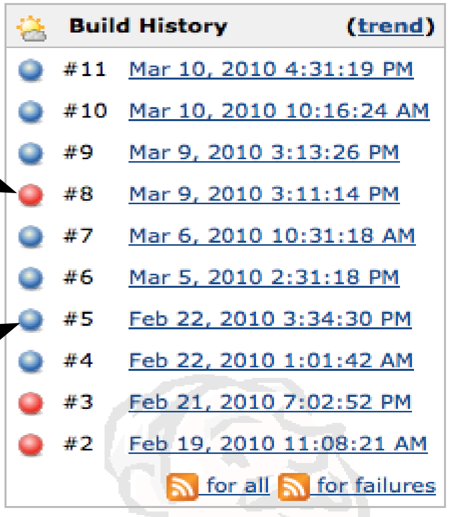
Hudson is ready to Build

That's all. Hudson is ready for Continuous integration. Automatically builds will get triggered when ever someone checked in to the SCM.

When a build is successful it is indicated by a blue ball. A red ball denotes a failed build.

Failed

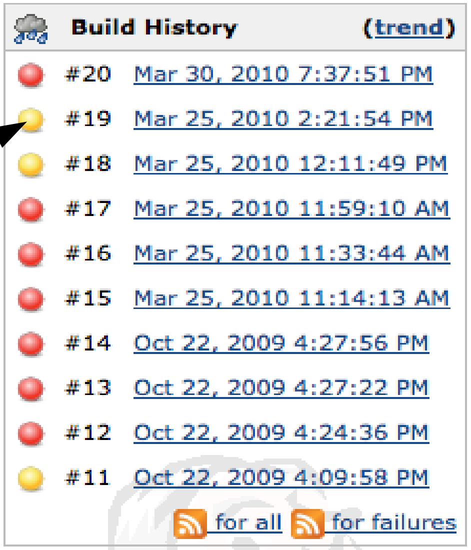
Successful



Build History		(trend)
🔵	#11	Mar 10, 2010 4:31:19 PM
🔵	#10	Mar 10, 2010 10:16:24 AM
🔵	#9	Mar 9, 2010 3:13:26 PM
🔴	#8	Mar 9, 2010 3:11:14 PM
🔵	#7	Mar 6, 2010 10:31:18 AM
🔵	#6	Mar 5, 2010 2:31:18 PM
🔵	#5	Feb 22, 2010 3:34:30 PM
🔵	#4	Feb 22, 2010 1:01:42 AM
🔴	#3	Feb 21, 2010 7:02:52 PM
🔴	#2	Feb 19, 2010 11:08:21 AM

for all for failures

Unstable

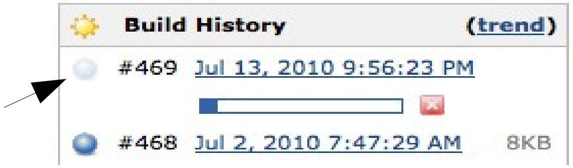


Build History		(trend)
🔴	#20	Mar 30, 2010 7:37:51 PM
🟡	#19	Mar 25, 2010 2:21:54 PM
🟡	#18	Mar 25, 2010 12:11:49 PM
🔴	#17	Mar 25, 2010 11:59:10 AM
🔴	#16	Mar 25, 2010 11:33:44 AM
🔴	#15	Mar 25, 2010 11:14:13 AM
🔴	#14	Oct 22, 2009 4:27:56 PM
🔴	#13	Oct 22, 2009 4:27:22 PM
🔴	#12	Oct 22, 2009 4:24:36 PM
🟡	#11	Oct 22, 2009 4:09:58 PM

for all for failures

Unstable (Ex. Failed test) builds are indicated by Yellow ball.

Fade Blinking

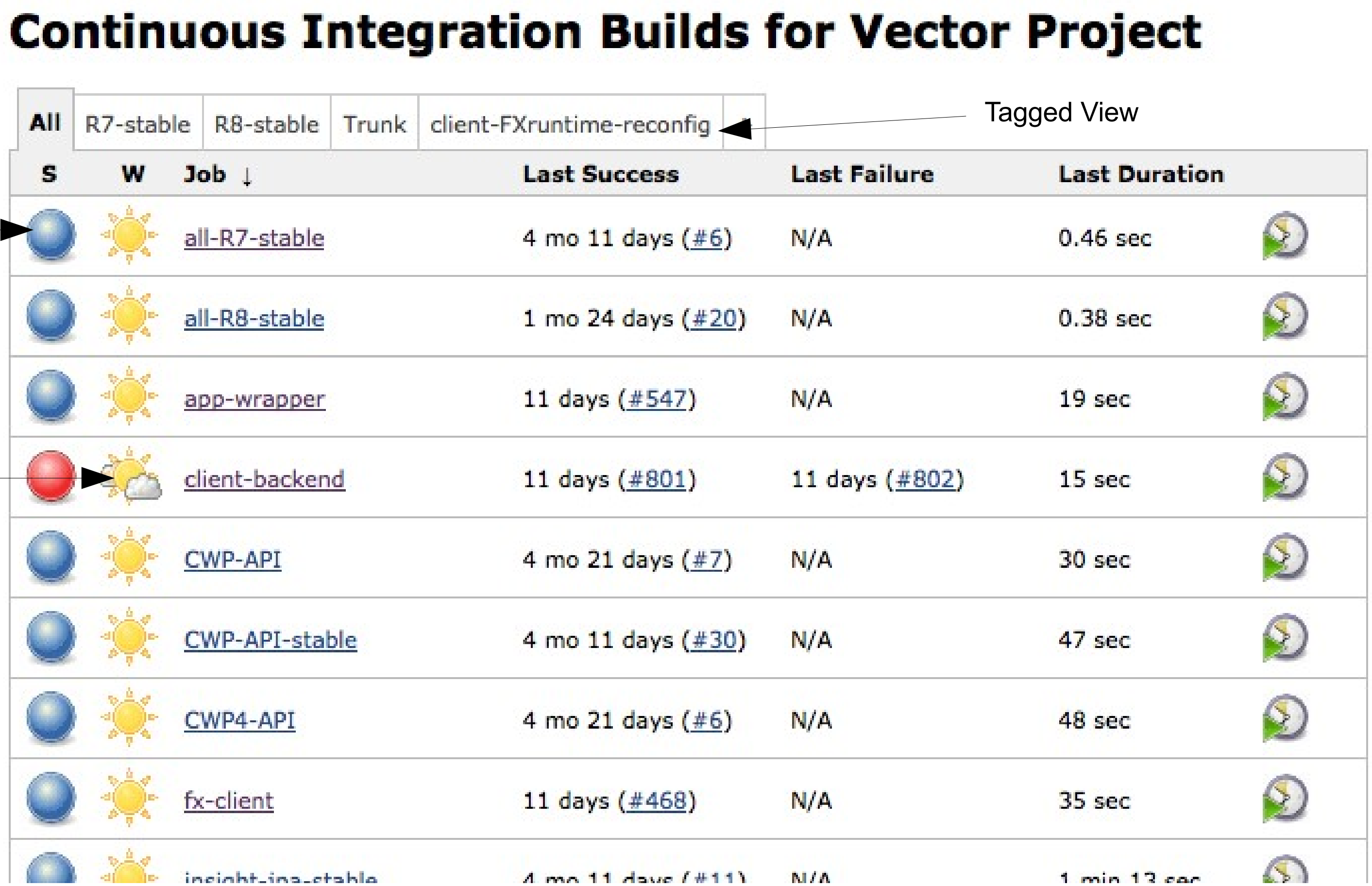


Build History		(trend)	
○	#469	Jul 13, 2010 9:56:23 PM	
🔵	#468	Jul 2, 2010 7:47:29 AM	8KB

If the ball is *blinking*, then it represents an ongoing build.

Hudson Main Dashboard

Hudson Main Dashboard provides a summary view of all the projects (jobs). Hudson also provide a way to tag the jobs to different views, so that it makes it easier to list the view by milestone or by other criteria.



Tagged View

Last Build Status

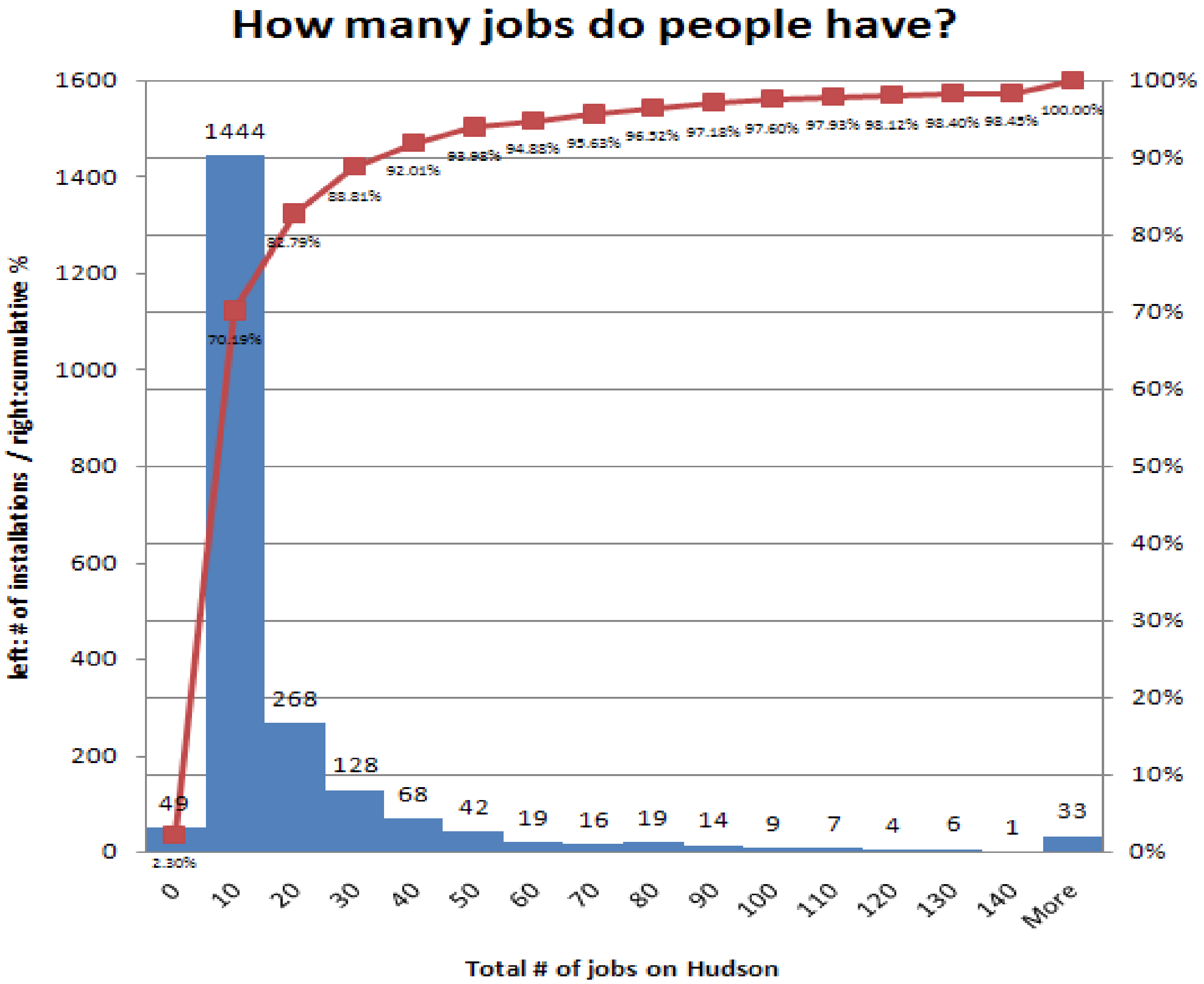
Overall Stability

Continuous Integration Builds for Vector Project

S	W	Job ↓	Last Success	Last Failure	Last Duration
●	☀	all-R7-stable	4 mo 11 days (#6)	N/A	0.46 sec
●	☀	all-R8-stable	1 mo 24 days (#20)	N/A	0.38 sec
●	☀	app-wrapper	11 days (#547)	N/A	19 sec
●	⛅	client-backend	11 days (#801)	11 days (#802)	15 sec
●	☀	CWP-API	4 mo 21 days (#7)	N/A	30 sec
●	☀	CWP-API-stable	4 mo 11 days (#30)	N/A	47 sec
●	☀	CWP4-API	4 mo 21 days (#6)	N/A	48 sec
●	☀	fx-client	11 days (#468)	N/A	35 sec
●	☀	insight-ipa-stable	4 mo 11 days (#11)	N/A	1 min 13 sec

How many Hudson jobs people are creating

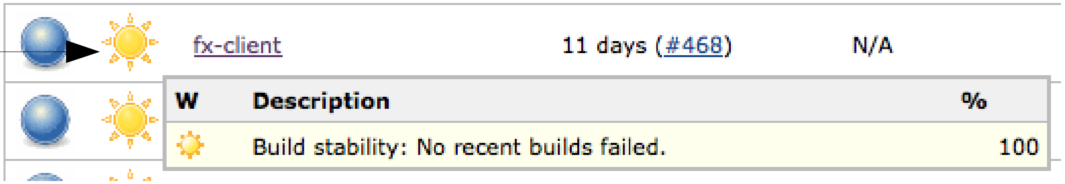
(based on 2009 survey)



90% of the Hudson installations run less than 30 jobs.

Continuous Integration Build Stability

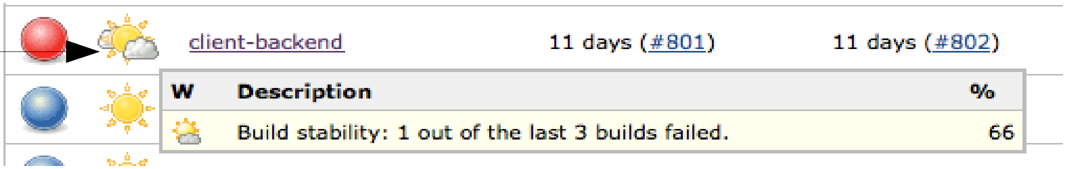
Highly Stable



fx-client 11 days (#468) N/A

W	Description	%
☀	Build stability: No recent builds failed.	100

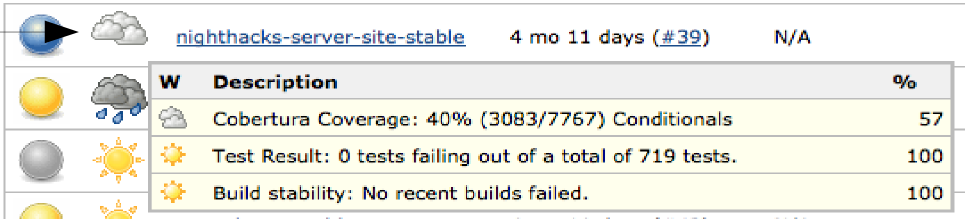
Slightly Unstable



client-backend 11 days (#801) 11 days (#802)

W	Description	%
⛅	Build stability: 1 out of the last 3 builds failed.	66

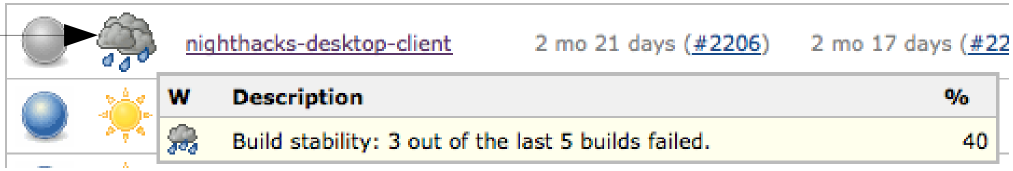
Unstable



nighthacks-server-site-stable 4 mo 11 days (#39) N/A

W	Description	%
☁	Cobertura Coverage: 40% (3083/7767) Conditionals	57
☀	Test Result: 0 tests failing out of a total of 719 tests.	100
☀	Build stability: No recent builds failed.	100

Highly Stable

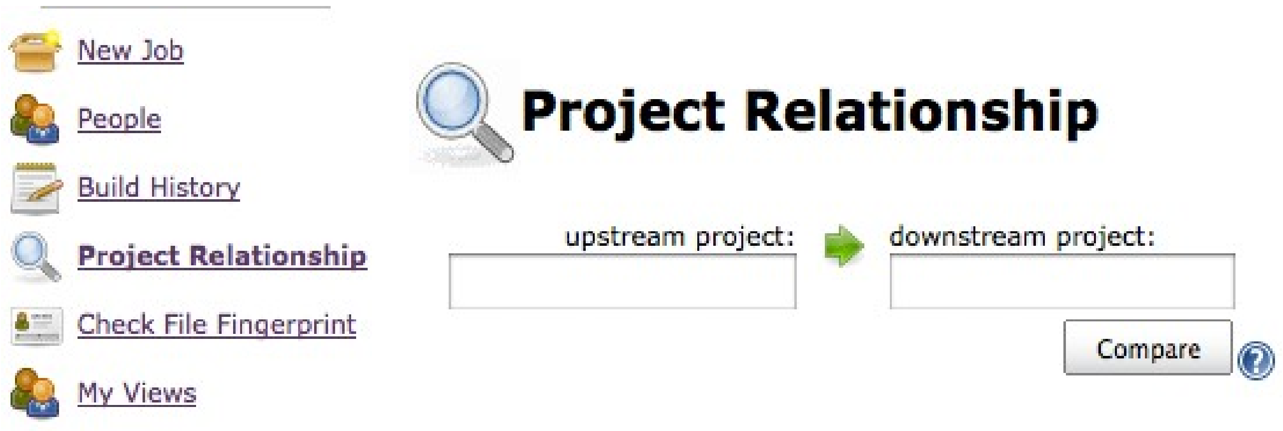


nighthacks-desktop-client 2 mo 21 days (#2206) 2 mo 17 days (#22

W	Description	%
🌧	Build stability: 3 out of the last 5 builds failed.	40

Project Relationship

When you have projects that depend on each other,Hudson can track which build of the upstream project is used by which build of the downstream project



The project relationship is accomplished by the conditions

- The upstream project records the fingerprints of its build artifacts
- The downstream project notes the fingerprints of the upstream jar files it uses

Tracking Versions using Fingerprints

The fingerprint of a file is simply a MD5 checksum. Hudson maintains a database of md5sum. For each md5sum, hudson maps it to a project and corresponding build. These files are stored at $HUDSON_HOME/fingerprints.

Project Relationship is maintained by

- jar files that your Upstream project produces.
- jar files that your dependent (downstream) project rely on.

Suppose there are two projects TOP and BOTTOM project and assume TOP depends on BOTTOM. You are working on the BOTTOM project. The TOP team reported that bottom.jar that they are using causes an NPE, which you thought you fixed in BOTTOM #32. Hudson can tell you which TOP builds are using (or not using) your bottom.jar #32 via fingerprints.

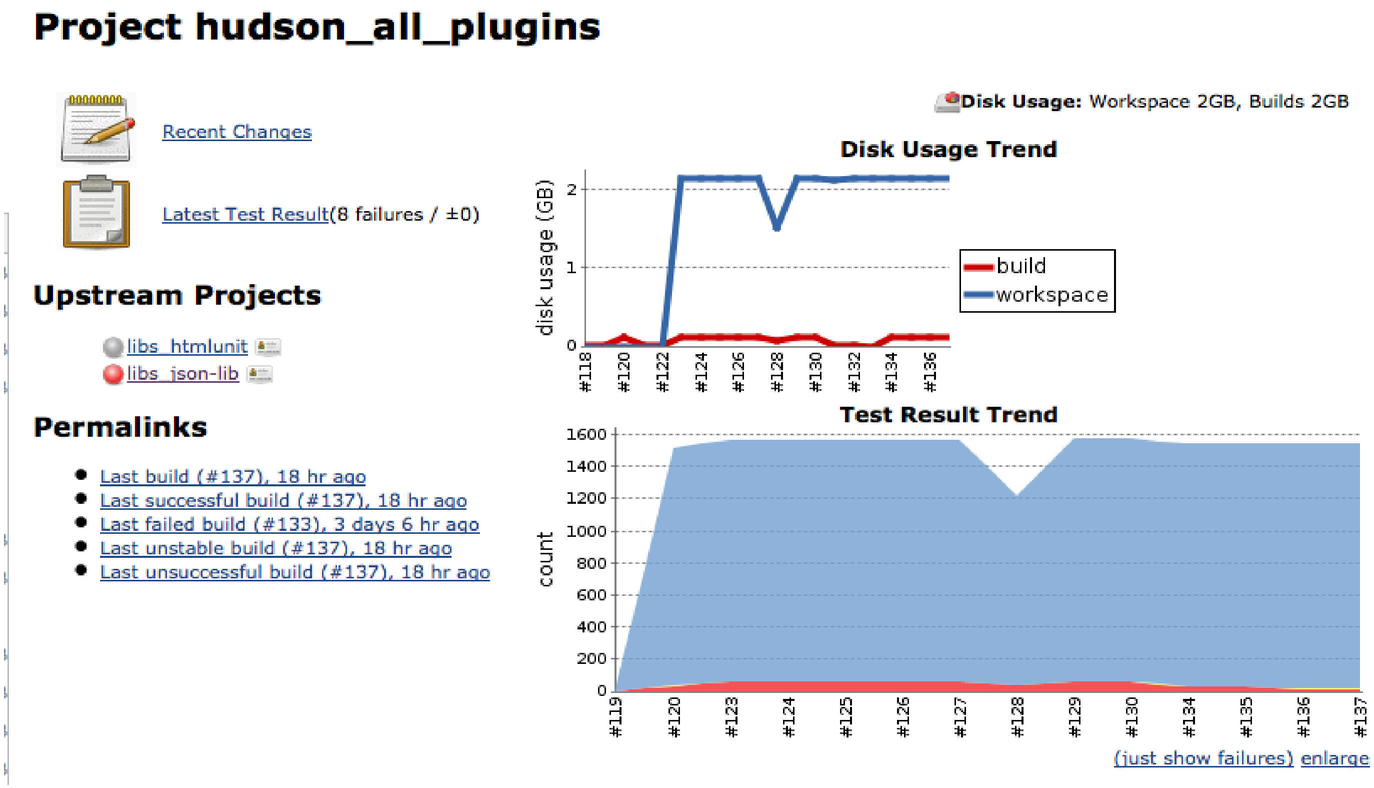
Project Dashboard

The Dashboard for particular project provides view for

- Last Successful Build Info
- Latest Test Result
- Monitoring Disk Usage
- Actions like configuring the job etc
- Test Result Trend
- Recent changes that caus

etc..

Various views in the project dashboard depends on various plugins installed.



Project hudson_all_plugins

Disk Usage: Workspace 2GB, Builds 2GB

Recent Changes

Latest Test Result(8 failures / ±0)

Upstream Projects

- libs_htmlunit
- libs_json-lib

Permalinks

- Last build (#137), 18 hr ago
- Last successful build (#137), 18 hr ago
- Last failed build (#133), 3 days 6 hr ago
- Last unstable build (#137), 18 hr ago
- Last unsuccessful build (#137), 18 hr ago

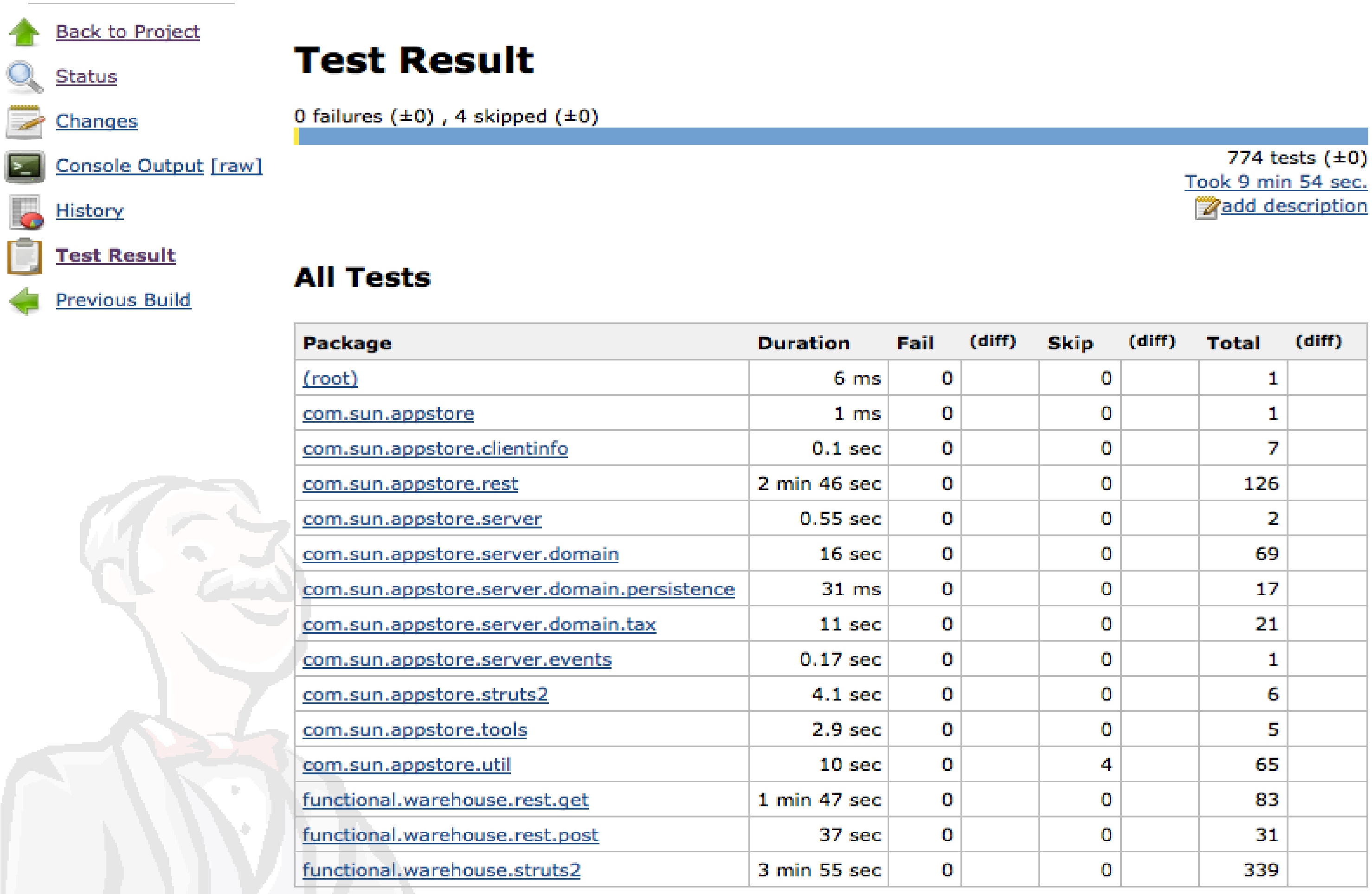
(just show failures) enlarge

Build Dashboard

The Dashboard for a particular Build provides view for

- Artifacts corresponding to this Build
- Changes that caused this Build
- Test Results
- Build console output

etc..



Test Result

0 failures (±0) , 4 skipped (±0)

774 tests (±0)
Took 9 min 54 sec.

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
(root)	6 ms	0		0		1	
com.sun.appstore	1 ms	0		0		1	
com.sun.appstore.clientinfo	0.1 sec	0		0		7	
com.sun.appstore.rest	2 min 46 sec	0		0		126	
com.sun.appstore.server	0.55 sec	0		0		2	
com.sun.appstore.server.domain	16 sec	0		0		69	
com.sun.appstore.server.domain.persistence	31 ms	0		0		17	
com.sun.appstore.server.domain.tax	11 sec	0		0		21	
com.sun.appstore.server.events	0.17 sec	0		0		1	
com.sun.appstore.struts2	4.1 sec	0		0		6	
com.sun.appstore.tools	2.9 sec	0		0		5	
com.sun.appstore.util	10 sec	0		4		65	
functional.warehouse.rest.get	1 min 47 sec	0		0		83	
functional.warehouse.rest.post	37 sec	0		0		31	
functional.warehouse.struts2	3 min 55 sec	0		0		339	

Distributed Building

Hudson supports the "master/slave" mode for distributed building.

Additional workload of building projects are delegated to multiple "slave" nodes

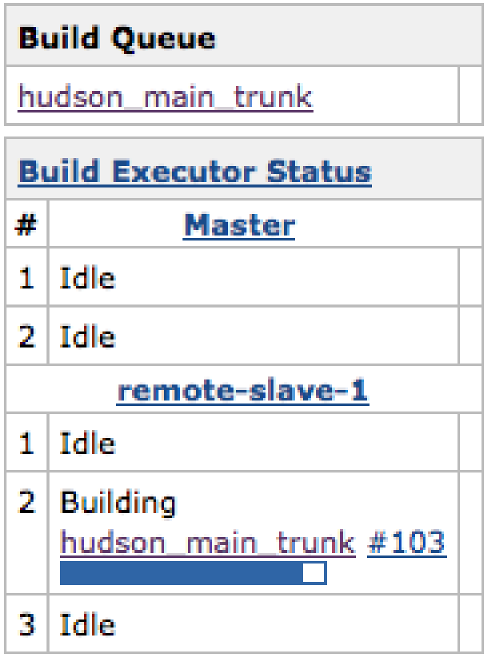
Provides different environments needed for builds/tests (Unix/Windows/Linux/Mac)

Master is an installation of Hudson. It serves all HTTP requests, and it also builds projects on its own.

Slaves are computers that are set up to build projects for a master. Hudson runs a separate program called **slave agent** on slaves. Master starts these slave agents on demand.

More details on Distributed Building is available at

http://wiki.hudson-ci.org/display/HUDSON/Distributed+builds



Build Queue

hudson_main_trunk

Build Executor Status

#	Master
1	Idle
2	Idle
	remote-slave-1
1	Idle
2	Building hudson_main_trunk #103
3	Idle

Popular Competitive Offerings

Apache Continuum — continuous integration server supporting Apache Maven and Apache Ant (open source)
Bamboo — commercial continuous integration server by Atlassian Software Systems
CruiseControl — Java-based framework for a continuous build process (open source)
TeamCity — commercial continuous-integration server by JetBrains.
Team Foundation Server — commercial continuous integration server and source code repository by Microsoft
Tinderbox — Mozilla-based product (open source)
Rational Team Concert — commercial software development collaboration platform by IBM