

Continuous Integration & Build Systems

Ryan Tidwell
Spring 2010
CSCI 5828

Continuous Integration & Build Systems

- Introduction
- Automated Build Tools
- Source Repositories
- Continuous Integration
- Organizing The Source Tree
- Automated Tests & Continuous Integration
- Continuous Integration Platforms
- Continuous Integration In Action

Introduction

- Continuous build systems are also referred to as “continuous integration” or CI systems.
- CI systems enable the continuous integration process
- Let's discuss some some fundamentals of continuous integration before looking at Hudson (a CI system)

Automated Build Tools

This class of tools provides developers a way to script the compilation of source code, thereby removing the complexity of compiling and packaging. Common build automation tools include make and ant.

Automated Build Tools

- As with all processes that we put in place to develop software, the process for building the system from source should be repeatable.
- Scripting builds with tools like make or ant will produce a build process that is easily repeated by individual developers.

Automated Build Tools

- This is great! We have a way to remove complexity from compiling large software systems.
- But how do we distribute the build script to developers? What if it changes? Where do developers even obtain the source that our build script will compile?

Source Repositories

- To solve these problems, we want to keep a single copy of the source code and build script so that we have a single repository for developers to work from.
- What if a large number of people need to make changes to the source in this repository? How should we handle conflicting changes in the repository?

Source Repositories

- We can manage access to the repository through systems such as CVS, SVN, and ClearCase.
- These systems allow developers to “check out” copies of the repository and manage modifications to the repository so that conflicts are avoided.
- Now we can ensure that everyone is seeing the same source and that any changes made to the source don't get lost.
- A single source repository is a key aspect of continuous integration.
- We want our CI system to build what is in the source repository. Builds can be triggered by check-ins or scheduled for regular execution.

Organizing The Source Tree

We want our builds to run quickly. Instead of building one giant executable, it is a best practice to organize the source so that the application is composed of smaller modules/sub-systems. This allows us to make changes to modules and sub-systems and build them quickly instead of waiting for the entire system to build in a developer sandbox every time we make a change

Continuous Integration

- Repeatable processes are easily automated by computers!
- We have a single repository for our source and a script for executing the build
- We now have the ability to introduce software that is capable of building our source as often as we want and publishing our final product as often as we want!

Continuous Integration Platforms

- Enter the CI system!
- CI systems provide a more sophisticated way of managing builds from our source repository
- These systems are more sophisticated than simple cron jobs
- CI systems provide us an environment for compiling, executing unit tests, and publishing the finished product
- Builds can be scheduled or done on-demand by responding to changes in the source repository
- All of this could theoretically be done from somebody's sandbox, so why bother?

Continuous Integration Platforms

- Processes that require a significant amount of human intervention are error prone. We need to have the build run perfectly EVERY time. In short, we need perfection!
- Using a CI system removes the human element from our build process and moves our process that much closer to perfection
- The CI system should be a part of our life-cycle

What Is a Continuous Build System?

Continuous build systems, also referred to as continuous integration systems (CI) automate the process of compiling and packaging source code. CI systems allow builds to happen in a single place where the whole team can get quick feedback on whether the code in the repository builds and unit tests are passing. CI systems also provide a way for the system to be installed after being built.

Examples Of Continuous Build Systems

Cruise Control
Cruise Control .NET
Hudson
Bamboo
Beebox
Apache Continuum

Tying It All Together

Now that we have a single source repository, modularized our system appropriately, and have introduced a CI system to build our code and execute test suites regularly, we now have the foundations of continuous integration in place.

Continuous Build Systems In Action

While there are many other powerful CI platforms such as Cruise Control, as a case study we will now look at Hudson in action within HP. Within HP's ESS Software division we employ Hudson as our continuous integration platform.

Continuous Build Systems In Action

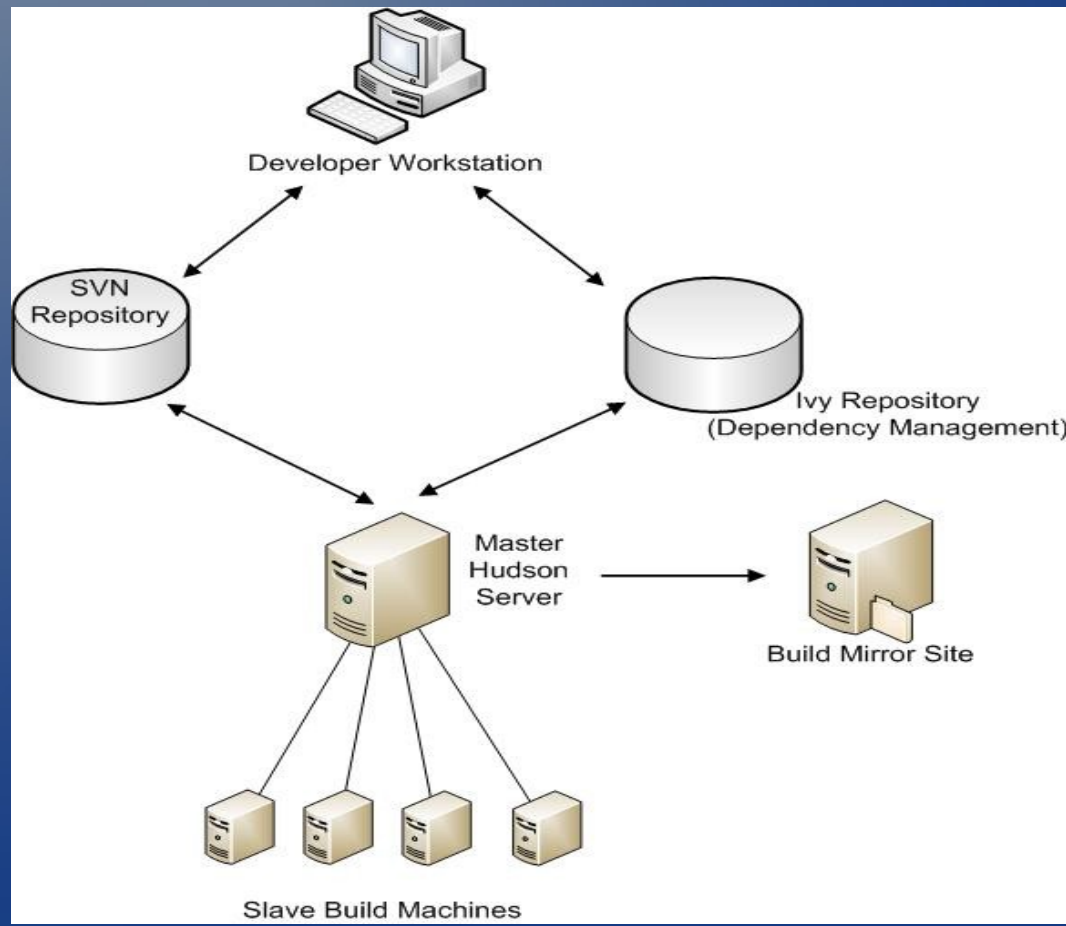
The software we develop (Insight Dynamics) is delivered as a suite composed of smaller products. We have decomposed our source tree into small modules that can be built individually.

Hudson builds each project in the suite using Ant and allows us to manage our build dependencies with Ivy.

Not only does Hudson build our source and run our test suites, it notifies us when builds fail and publishes the test results.

Continuous Build Systems In Action

Our build environment makes use of the distributed build management capabilities of Hudson. We run builds on Windows, HP-UX, and Linux Hudson slaves so that we can build customized executables for Windows, HP-UX, and Linux.



Continuous Build Systems In Action

Here are some screenshots from Hudson. Hudson installs easily on Windows, Linux, HP-UX, and a variety of other operating systems. It is implemented in Java and is therefore platform independent. There are hundreds of plugins available for Hudson. By default, Hudson comes with Ant, Maven, and CVS plugins installed. Hudson can build projects written in virtually any language. Development teams also have the flexibility to develop their own custom Hudson plugins for further customizing their build process.

Continuous Build Systems In Action

The screenshot shows the Hudson web interface. The browser address bar is <http://localhost:8080/>. The page title is "Hudson". On the left sidebar, there are links for "New Job", "Manage Hudson", "People", and "Build History". Below these are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (2 idle executors). The main content area displays a table of jobs. The job "Test1" is shown with a success icon (sun) and a last success time of "1 day 1 hr (#1)". A tooltip for "Test1" shows a "W" (warning) icon and a "Build stability: No recent builds failed." with a "100%" success rate. On the right, there are links for "ENABLE AUTO REFRESH" and "add description". At the bottom right, there are RSS feeds for "Legend", "for all", "for failures", and "for just latest builds".

Build Queue
No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

S	W	Job ↓	Last Success	Last Failure	Last Duration
		Test1	1 day 1 hr (#1)	N/A	0.42 sec

Icon: [S](#) [M](#) [L](#)

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

[Legend](#) [for all](#) [for failures](#) [for just latest builds](#)

Continuous Build Systems In Action

Hudson

search ?

ENABLE AUTO REFRESH

-  [New Job](#)
-  [Manage Hudson](#)
-  [People](#)
-  [Build History](#)
-  [Install as Windows Service](#)

Manage Hudson



[Configure System](#)
Configure global settings and paths.



[Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.



[Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Hudson.



[System Information](#)
Displays various environmental information to assist trouble-shooting.



[System Log](#)
System log captures output from `java.util.logging` output related to Hudson.



[Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



[Hudson CLI](#)
Access/manage Hudson from your shell, or from your script.



[Script Console](#)
Executes arbitrary script for administration/trouble-shooting/diagnostics.



[Manage Nodes](#)
Add, remove, control and monitor the various nodes that Hudson runs jobs on.



[Install as Windows Service](#)
Installs Hudson as a Windows service to this system, so that Hudson starts automatically when the machine boots.



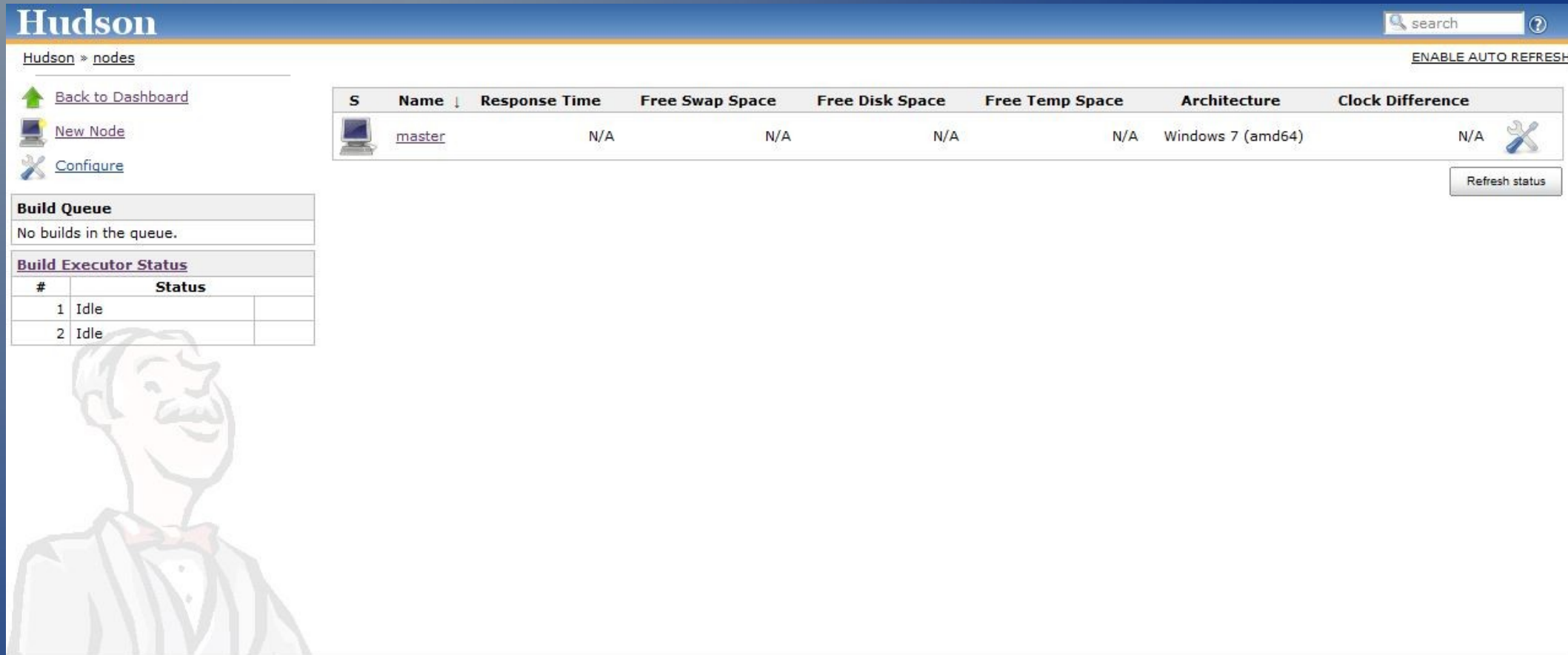
[Get Support Subscription](#)
Commercial support subscription available from Sun Microsystems.

Build Queue
No builds in the queue.

Build Executor Status	
#	Status
1	Idle
2	Idle



Continuous Build Systems In Action





The screenshot shows the Hudson web interface. At the top left is the 'Hudson' logo. To its right is a search bar and a help icon. Below the logo is a breadcrumb trail 'Hudson > nodes' and a link to 'ENABLE AUTO REFRESH'. On the left sidebar, there are links for 'Back to Dashboard', 'New Node', and 'Configure'. The main content area features a table of nodes with columns for 'S', 'Name', 'Response Time', 'Free Swap Space', 'Free Disk Space', 'Free Temp Space', 'Architecture', and 'Clock Difference'. A single node named 'master' is listed with 'N/A' values for response, swap, disk, and temp space, and 'Windows 7 (amd64)' for architecture. A 'Refresh status' button is located to the right of the table. Below the table, there are two sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing a table with two executors, both in an 'Idle' state. A faint cartoon illustration of a man with a mustache and a bow tie is visible in the bottom left corner of the interface.

Hudson ?

Hudson > nodes ENABLE AUTO REFRESH

[Back to Dashboard](#)
[New Node](#)
[Configure](#)

S	Name ↓	Response Time	Free Swap Space	Free Disk Space	Free Temp Space	Architecture	Clock Difference
	master	N/A	N/A	N/A	N/A	Windows 7 (amd64)	N/A 

Build Queue
No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Other Resources

Martin Fowler on continuous integration:

<http://martinfowler.com/articles/continuousIntegration.html>

Hudson Home (Allows you to take Hudson for a test drive before installing!):

<http://hudson-ci.org/>

Wikipedia:

http://en.wikipedia.org/wiki/Continuous_integration

“Continuous Integration in the cloud with Hudson”:

http://java.sun.com/javaone/2009/articles/gen_hudson.jsp

Official Hudson blog:

<http://blog.hudson-ci.org/>