

# Jenkins, votre serviteur

C. Loomis (CNRS/LAL)

Journée LoOPS

11 décembre 2012



# Intégration Continue : Wikipedia

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

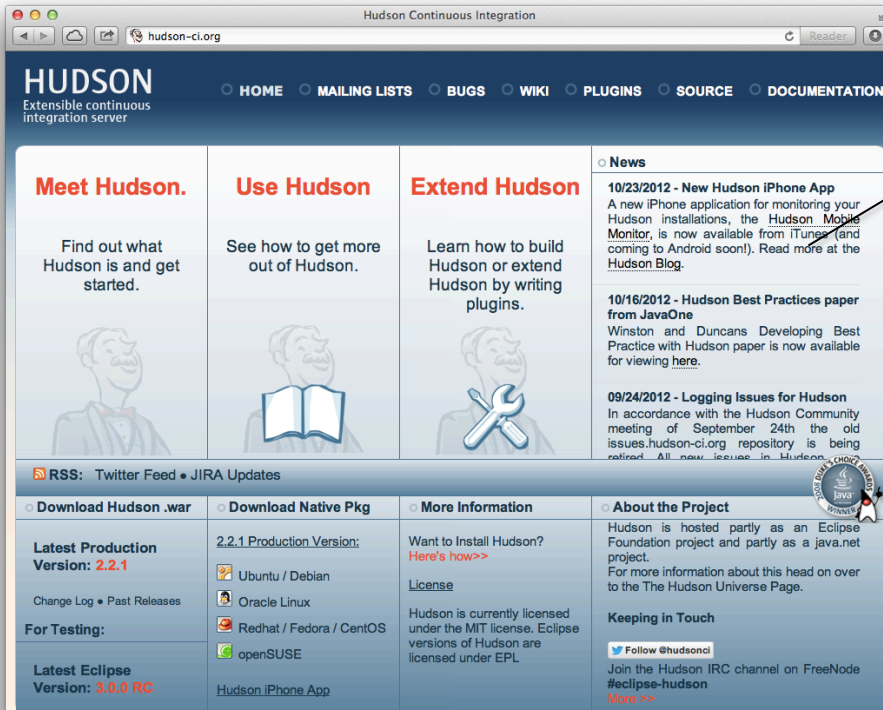
# Intégration Continue : Pratique

- Etre plus confiant que un logiciel fonctionne correctement et efficacement
  - Améliorer la qualité du code
  - Vérifier le comportement du code avec les tests unitaires
  - Déployer un système complet et valider la fonctionnalités
- Tous pour chaque changement du code!

# Les Outils Automatisés

- L'intégration continue est possible avec un minimum d'effort humain grâce à des outils automatisés
- Open Source
  - Hudson, Jenkins, Travis-CI, CruiseControl, ...
- Commerciaux
  - Bamboo, TeamCity, ...

# Hudson & Jenkins



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



Jenkins  
<http://jenkins-ci.org>

# Le « Fork » Hudson/Jenkins

- L'existence des deux produits quasi-identiques est un (des plusieurs) effet de bord de l'acquisition du Sun par Oracle
  - La communauté open source (responsable pour un grand nombre de plugins) a créé Jenkins
  - Oracle/Sonatype continue le développement du Hudson, mais dans l'Eclipse Foundation
- Malheureusement trop tard pour fusionner les communautés au tour d'un produit unique

# StratusLab

The screenshot shows the Hudson dashboard interface. The browser address bar displays 'https://hudson.stratuslab.eu'. The main content area features a table of build jobs with columns for status (S), weather icon (W), job name (Job), and last success time. A sidebar on the left contains navigation links and a 'Build Queue' section.

All	Build_CentOS	Build_OpenSuSE	CentOS_snapshots	Certification	Install	Marketplace	OpenNebula	Registration	Release_CentOS
S	W	Job ↓	Last Success						
●	☁	<a href="#">authn_Release</a>	7 mo 11 days (#27)						
●	☀	<a href="#">benchmarks_Release</a>	9 mo 15 days (#14)						
●		<a href="#">biocomp_update_weekly</a>	N/A						
●	☀	<a href="#">build_ALL_CentOS</a>	14 hr (#242)						
●	☀	<a href="#">build_ALL_OpenSuSE</a>	14 hr (#131)						
●	☀	<a href="#">build_authn_CentOS</a>	14 hr (#399)						
●	☀	<a href="#">build_authn_OpenSuSE</a>	14 hr (#220)						
●	☀	<a href="#">build_benchmarks_CentOS</a>	14 hr (#260)						
●	☀	<a href="#">build_benchmarks_OpenSuSE</a>	14 hr (#176)						
●	☁	<a href="#">build_client_CentOS</a>	3 hr 6 min (#598)						
●	☁	<a href="#">build_client_OpenSuSE</a>	3 hr 6 min (#348)						
●	☀	<a href="#">build_distribution_CentOS</a>							
●	☀	<a href="#">build_distribution_OpenSuSE</a>							
●	☀	<a href="#">build_image-recipes_CentOS</a>	14 hr (#91)						
●	☀	<a href="#">build_image-recipes_OpenSuSE</a>	8 days 1 hr (#17)						

<http://hudson.stratuslab.eu/>

# Quattor

The screenshot displays the Jenkins web interface for the Quattor project. The browser address bar shows <https://jenkins1.ugent.be/view/Quattor/>. The Jenkins logo is at the top left, and a search bar and 'log in' link are at the top right. The main content area has tabs for 'All', 'EasyBuild', and 'Quattor'. The 'Jobs Grid' table lists various jobs with their status icons (green for success, red for failure, blue for in progress). Below the grid is a 'Latest builds' table showing the most recent builds for each job. To the right, a 'Test Statistics Chart' shows a pie chart with 100% success. A 'Test Trend Chart' is partially visible at the bottom right. A URL box is overlaid on the bottom right of the screenshot.

Job	Build	Time
<a href="#">NCM_components</a>	<a href="#">#222</a>	Dec 10, 2012 12:55:11 AM
<a href="#">ncm-query</a>	<a href="#">#37</a>	Dec 10, 2012 12:40:25 AM
<a href="#">CCM</a>	<a href="#">#118</a>	Dec 10, 2012 12:38:44 AM
<a href="#">perl-CAF</a>	<a href="#">#97</a>	Dec 10, 2012 12:38:03 AM
<a href="#">NCM_components</a>	<a href="#">#221</a>	Dec 10, 2012 12:37:54 AM
<a href="#">perl-LC</a>	<a href="#">#91</a>	Dec 10, 2012 12:37:36 AM
<a href="#">ncm-ncd</a>	<a href="#">#90</a>	Dec 10, 2012 12:37:23 AM
<a href="#">ncm-cdispd</a>	<a href="#">#112</a>	Dec 10, 2012 12:37:12 AM
<a href="#">cdp-listend</a>	<a href="#">#109</a>	Dec 10, 2012 12:36:48 AM
<a href="#">AII</a>	<a href="#">#73</a>	Dec 10, 2012 12:36:48 AM

#	Status
1	Idle
2	Idle

Test Statistics Chart:

- skipped = 0 (0%)
- failed = 0 (0%)
- successes = 1,700 (100%)

Test Trend Chart:

count

<http://jenkins1.ugent.be/>



# SixSq

The screenshot shows the Jenkins dashboard for 'ci.sixsq.com'. The main content is a table of builds with columns for status (S), weather icon (W), name, last success, last failure, and last duration. The table lists various build jobs like EBU-TTF, SlipStream\_deploy\_new, and SlipStreamClient\_build. On the left, there are navigation links for 'New Job', 'People', 'Build History', etc., and a 'Build Queue' section showing 'No builds in the queue'. At the bottom, there are links for 'Help us localize this page' and 'Page generated: Dec 10, 2012 5:09:15 PM Jenkins ver. 1.466.2'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">EBU-TTF</a>	3 hr 19 min (#65)	7 days 6 hr (#59)	20 min
		<a href="#">EBU-TTF-Release</a>	N/A	5 days 23 hr (#2)	7 min 38 sec
		<a href="#">SlipStream_deploy_new</a>	9 days 0 hr (#21)	9 days 1 hr (#20)	10 min
		<a href="#">SlipStream_deploy_update</a>	N/A	N/A	N/A
		<a href="#">SlipStream_selenium_tests</a>	1 yr 0 mo (#8)	1 mo 14 days (#24)	1 min 3 sec
		<a href="#">SlipStreamClient_build</a>	28 days (#111)	8 days 5 hr (#112)	1 min 57 sec
		<a href="#">SlipStreamDoc_build</a>	8 days 5 hr (#8)	8 days 5 hr (#10)	4 min 1 sec
		<a href="#">SlipStreamMta_build</a>	8 days 5 hr (#64)	1 mo 12 days (#4)	2 min 22 sec
		<a href="#">SlipStreamParent_build</a>	8 days 5 hr (#10)	8 days 5 hr (#9)	4 min 58 sec
		<a href="#">SlipStreamServer_build</a>	12 days (#789)	5 days 21 hr (#796)	14 min
		<a href="#">SlipStreamServerBackup_build</a>	8 days 5 hr (#92)	1 mo 14 days (#3)	1 min 28 sec

# StratusLab

- Projet européen et maintenant communauté ouverte pour la développement d'une un logiciel cloud « IaaS »
- Utiliser Hudson pour 2.5+ ans pour automatiser les tâches de développement
  - Testing Continu
  - Intégration Continue
  - Gestion des Releases

# Vocabulaire

The screenshot shows the Hudson dashboard with the following components:

- Dashboard [Hudson]**: Title bar at the top.
- Views**: A row of tabs at the top of the main content area, including 'All', 'Build\_CentOS', 'Build\_OpenSuSE', 'CentOS\_snapshots', 'Certification', 'Install', 'Marketplace', 'OpenNebula', 'Registration', and 'Release\_CentOS'. The 'All' tab is selected.
- Jobs Table**: A table listing various jobs with columns for status (green or grey circle), weather icon, job name, and last success time. Jobs include 'authn\_Release', 'benchmarks\_Release', 'biocomp\_update\_weekly', 'build\_ALL\_CentOS', 'build\_ALL\_OpenSuSE', 'build\_authn\_CentOS', 'build\_authn\_OpenSuSE', 'build\_benchmarks\_CentOS', 'build\_benchmarks\_OpenSuSE', 'build\_client\_CentOS', 'build\_client\_OpenSuSE', 'build\_distribution\_CentOS', 'build\_distribution\_OpenSuSE', 'build\_image-recipes\_CentOS', and 'build\_image-recipes\_OpenSuSE'.
- Build Queue**: A section on the left sidebar showing 'No builds in the queue.'
- Build Executor Status**: A section on the left sidebar showing the status of various executors, including 'Master 0/2' and several slave nodes like 'cloud-centos62\_1 0/2', 'cloud-centos62\_2 offline', 'cloud-centos62\_3 offline', 'cloud-centos62-client\_1 0/1', 'cloud-centos62-client\_2 offline', 'cloud-lal-opensuse12 (vm-88) 0/2', 'GRNet-CentOS-1 offline', 'GRNet-Fedora14-2 offline', 'GRNET-hudson-cert-1 0/1', 'GRNET-hudson-cert-2 offline', and 'GRNet-Ubuntu-1 offline'.

Views : Groupes des tâches similaires

Jobs : Tâches automatisés

Nodes : Esclaves pour exécuter les tâches

# Définition d'un Job

The screenshot displays the Hudson configuration page for a job named 'build\_client\_CentOS'. The interface includes a sidebar with navigation links such as 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Dependency Graph', and 'Git Polling Log'. The main configuration area is divided into several sections:

- Project name:** build\_client\_CentOS
- Cascading Project:** None
- Description:** (Empty text area)
- Discard Old Builds:**
- This build is parameterized:**
- Github project:** (Empty text field)
- Disable Build (No new builds will be executed until the project is re-enabled.):**
- Execute concurrent builds if necessary (beta):**
- Restrict where this project can be run:**
- Node and label menu:** Node: centos-6.2 (group of cloud-centos62\_1, cloud-centos62\_2, cloud-centos62\_3)
- Advanced Node and Label expressions:**
- Advanced Project Options:** (Advanced... button)
- Source Code Management:**  Git
- Repositories:** URL of repository: git://github.com/StratusLab/client.git (Advanced... button)
- Branches to build:** Branch Specifier (blank for default): master (Add, Delete Branch, Advanced... buttons)

On the left, the **Build History** table shows a list of builds with their IDs, dates, and times:

#	Date	Time
#599	Dec 11, 2012	4:05:04 AM
#598	Dec 10, 2012	3:00:25 PM
#597	Dec 10, 2012	9:15:25 AM
#596	Dec 10, 2012	4:05:00 AM
#595	Dec 9, 2012	7:40:25 AM
#594	Dec 9, 2012	4:04:58 AM
#593	Dec 8, 2012	4:05:16 AM
#592	Dec 7, 2012	3:00:25 PM
#591	Dec 7, 2012	2:50:25 PM
#590	Dec 7, 2012	4:05:00 AM
#589	Dec 6, 2012	9:40:25 PM
#588	Dec 6, 2012	3:20:25 PM
#587	Dec 6, 2012	4:05:00 AM
#586	Dec 5, 2012	4:05:10 AM
#585	Dec 4, 2012	4:04:55 AM
#584	Dec 3, 2012	8:50:25 PM
#583	Dec 3, 2012	9:15:25 AM

# Définition d'un Job

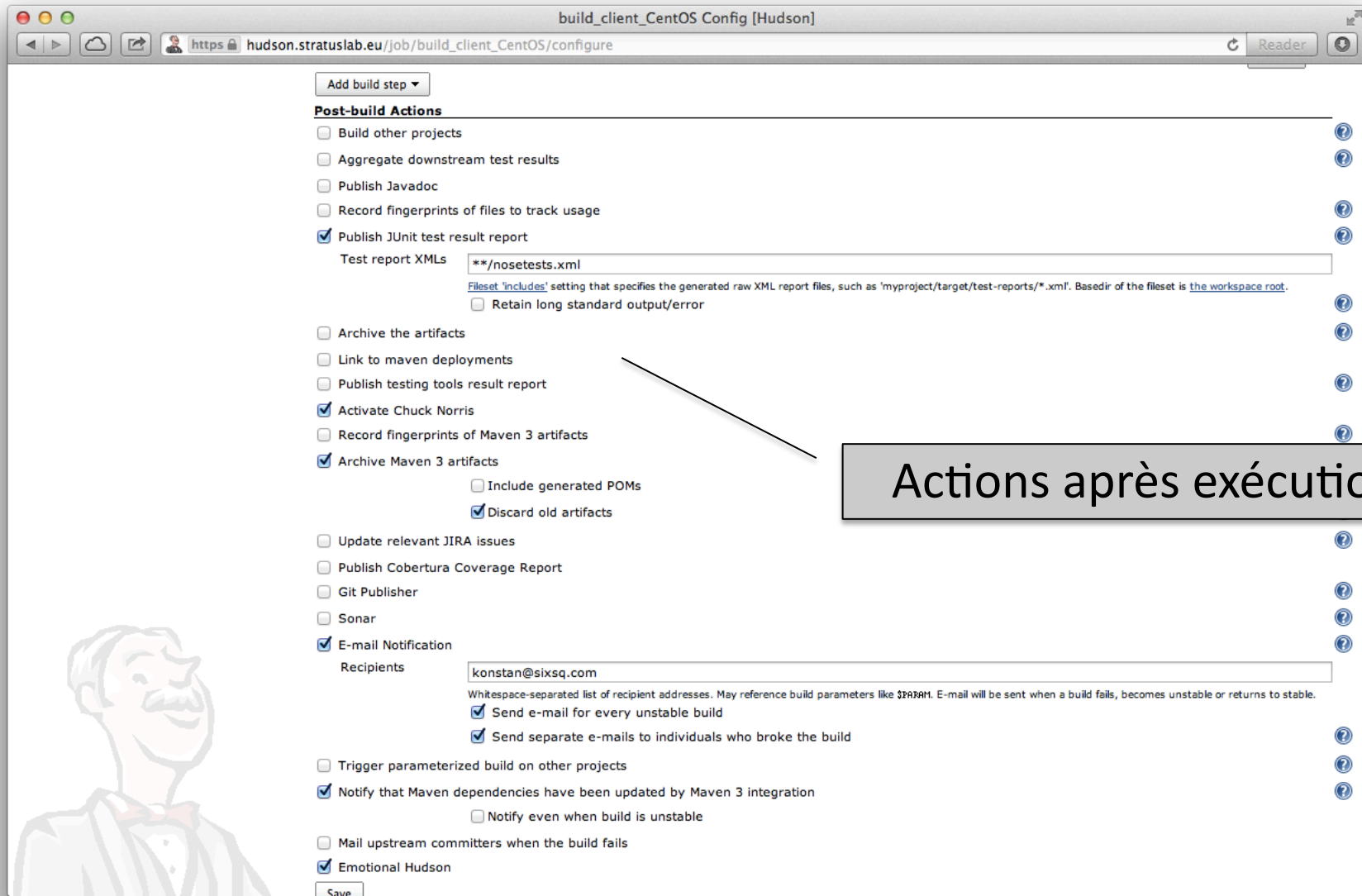
The screenshot shows the Jenkins configuration page for a job named 'build\_client\_CentOS'. The page is divided into several sections:

- Build Triggers:** Contains options for when to build. A callout box labeled 'Déclenchement du job' points to the 'Build after other projects are built' checkbox, which is checked. Below it, the 'Projects names' field contains 'build\_metadata\_CentOS'. Other options include 'Trigger builds remotely', 'Build periodically', 'Poll SCM' (checked), and 'Build when Maven dependencies have been updated'.
- Build:** Contains the 'Execute shell' step. A callout box labeled 'Procédure(s)' points to the 'Command' field, which contains a shell script:

```
# Explicitly remove firewall test so this doesn't need to run as root.
#
export PYTHONPATH=$WORKSPACE/core/src/main/python
cd $WORKSPACE/core/test
pwd
nosetests --verbose --with-xunit $(find . -name "*Test*.py" -a -not -name "*Firewall*.py")
```
- Invoke Maven 3:** Contains the 'Maven 3' dropdown (set to '(Bundled)'), 'Goals' (set to 'clean deploy'), and 'Properties' field.
- Post-build Actions:** Contains the 'Build other projects' checkbox.

On the left side of the page, there is a list of recent build runs with their IDs and timestamps. At the bottom, there are buttons for 'Add build step', 'Advanced', and 'Delete'.

# Définition d'un Job



build\_client\_CentOS Config [Hudson]

https://hudson.stratuslab.eu/job/build\_client\_CentOS/configure

Add build step ▾

**Post-build Actions**

- Build other projects
- Aggregate downstream test results
- Publish Javadoc
- Record fingerprints of files to track usage
- Publish JUnit test result report
  - Test report XMLs:
  - [Fileset 'includes'](#) setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/\*.xml'. Basedir of the fileset is [the workspace root](#).
  - Retain long standard output/error
- Archive the artifacts
- Link to maven deployments
- Publish testing tools result report
- Activate Chuck Norris
- Record fingerprints of Maven 3 artifacts
- Archive Maven 3 artifacts
  - Include generated POMs
  - Discard old artifacts
- Update relevant JIRA issues
- Publish Cobertura Coverage Report
- Git Publisher
- Sonar
- E-mail Notification
  - Recipients:
  - Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM.
  - Send e-mail for every unstable build
  - Send separate e-mails to individuals who broke the build
- Trigger parameterized build on other projects
- Notify that Maven dependencies have been updated by Maven 3 integration
  - Notify even when build is unstable
- Mail upstream committers when the build fails
- Emotional Hudson

Save

Actions après exécution

# Vocabulaire

The screenshot shows the Hudson Update Center interface. The browser address bar indicates the URL <https://hudson.stratuslab.eu/pluginManager/installed>. The page title is "Hudson" and the breadcrumb is "Hudson » Plugin Manager". There is a search bar and links for "loomis" and "log out".

The main content area displays a table of installed plugins. The table has columns for "Enabled", "Name", "Version", "Previously installed version", and "Pinned". The "Name" column is sorted in descending order. The table lists various plugins, some of which are checked as "Enabled".

Enabled	Name ↓	Version	Previously installed version	Pinned
<input checked="" type="checkbox"/>	<a href="#">Blame Upstream Committers</a> A simple plugin that allows downstream jobs to mail upstream committers when the build fails.	1.2		
<input checked="" type="checkbox"/>	<a href="#">ChuckNorris Plugin</a> ChuckNorris plugin displays a picture of Chuck Norris (instead of Hudson the butler) and a random Chuck Norris 'The Programmer' fact on each build page.	0.4		
<input checked="" type="checkbox"/>	<a href="#">Cobertura Plugin</a> This plugin integrates <a href="#">Cobertura coverage reports</a> to Hudson.	1.1		
<input type="checkbox"/>	<a href="#">Hudson CVS Plug-in</a>	2.2.0		
<input checked="" type="checkbox"/>	<a href="#">Hudson Dependency Analyzer Plugin</a> This plugin search for the dependency:analyze results into the maven build output and summarize it.	0.5		
<input checked="" type="checkbox"/>	<a href="#">Dependency Graph Viewer Plugin</a> This plugin shows a dependency graph of the projects. It uses dot (from graphviz) for drawing.	0.2	Downgrade to 0.1	
<input type="checkbox"/>	<a href="#">Hudson Amazon EC2 plugin</a> This plugin integrates Hudson with <a href="#">Amazon EC2</a> or anything implementing the EC2 API's such as an Ubuntu.	1.10		
<input checked="" type="checkbox"/>	<a href="#">Emotional Hudson plugin</a> This funny plugin changes the expression of Mr. Hudson in the background when your builds fail.	1.2		
<input checked="" type="checkbox"/>	<a href="#">Hudson GIT plugin</a> This plugin integrates <a href="#">GIT</a> with Hudson.	2.2.1	Downgrade to 2.2.0	Unpin ?
<input checked="" type="checkbox"/>	<a href="#">Github plugin</a> This plugin integrates <a href="#">Github</a> to Jenkins.	0.4		
<input checked="" type="checkbox"/>	<a href="#">Green Balls</a> Because green is better than blue!	1.10		
<input checked="" type="checkbox"/>	<a href="#">Hudson iPhoneView plugin</a> This plugin allows you to view the status of your jobs via iPhone or iPod touch.	0.2		

A callout box with a grey background and black border contains the text: "Plugins : Composants pour gérer les interactions avec les services externes". A line points from this box to the table of plugins.

# Vocabulaire

Trend : Histoire des résultats des tests unitaires

**Project build\_client\_CentOS**

Workspace

Artifact(s) of the Last Successful Build

- stratuslab-cli-sysadmin-rpm-2.8-SNAPSHOT.rpm
- stratuslab-cli-sysadmin-zip-2.8-SNAPSHOT.zip
- stratuslab-cli-user-pkg-2.8-SNAPSHOT.tar.gz
- stratuslab-cli-user-pkg-2.8-SNAPSHOT.zip
- stratuslab-cli-user-rpm-2.8-SNAPSHOT.rpm
- stratuslab-core-zip-2.8-SNAPSHOT-core-bundle.tar.gz
- stratuslab-core-zip-2.8-SNAPSHOT-core-bundle.zip

Recent Changes

Latest Test Results (no failures)

Latest Console output

Latest Maven Build Information

Upstream Projects

- build\_metadata\_CentOS

**Test Result Trend**

count

100

80

60

40

20

0

Chuck Norris doesn't believe in floating point numbers because they can't be typed on his binary keyboard.

(just show failures) enlarge

Build Number	Timestamp	Status
#599	Dec 11, 2012 4:05:04 AM	Success
#598	Dec 10, 2012 3:00:25 PM	Success
#597	Dec 10, 2012 9:15:25 AM	Success
#596	Dec 10, 2012 4:05:00 AM	Success
#595	Dec 9, 2012 7:40:25 AM	Success
#594	Dec 9, 2012 4:04:58 AM	Failure
#593	Dec 8, 2012 4:05:16 AM	Failure
#592	Dec 7, 2012 3:00:25 PM	Failure
#591	Dec 7, 2012 2:50:25 PM	Failure
#590	Dec 7, 2012 4:05:00 AM	Success
#589	Dec 6, 2012 9:40:25 PM	Success
#588	Dec 6, 2012 3:20:25 PM	Success
#587	Dec 6, 2012 4:05:00 AM	Success
#586	Dec 5, 2012 4:05:10 AM	Success

Artifacts : Le code packagé

Histoire : Les résultats des jobs avec une identifier



# Testing Continu

- Exécuter les tests unitaires sur plusieurs OS après chaque commit
  - Bon intégration avec les gestionnaires du code : git, svn, cvs, ...
  - Définir une matrice des environnement différentes : OS, java, ...
  - Les jobs peuvent être déclencher automatiquement avec notification des résultats
  - Supporte excellente pour maven

# Contrôle de Qualité

- Pour StratusLab est intégré avec les tests unitaires et automatiser au maximum
- FindBugs – analyseur statique pour Java
  - Traiter comme un test unitaire pour les composants StratusLab écrit en Java
  - Ça trouve les problèmes de codage fréquents et forme les développeurs non-experts en Java
  - Les « code reviews » peuvent concentrer sur les problèmes de design et architecture

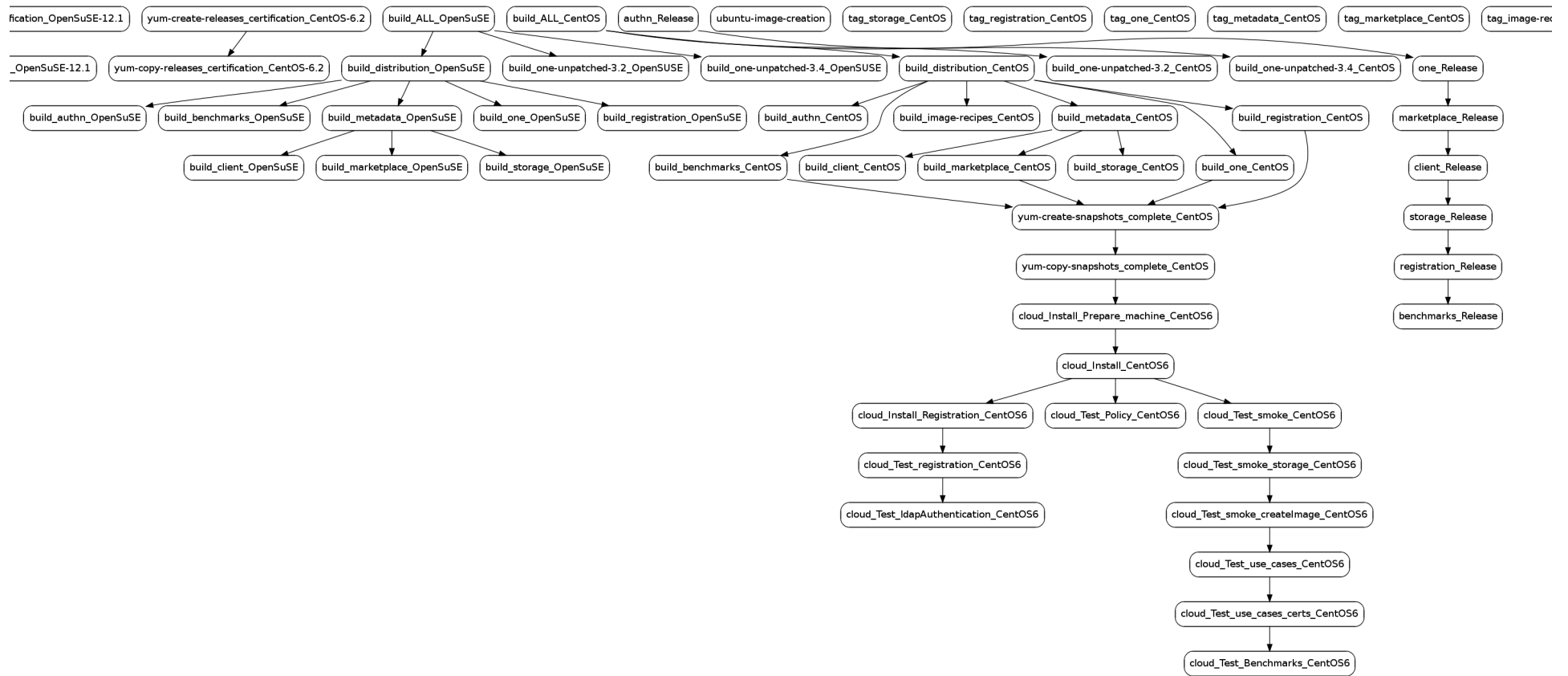
# Travis-CI

- Après son passage à GitHub, StratusLab utilise aussi Travis-CI pour le testing continu
  - + Les tests sont déclencher automatiquement avec les commits dans GitHub
  - + On peut afficher un badge « pass, fail » dans le README du projet
  - + Rien a déployer : les esclaves sont déjà là
  - Les environnements disponibles sont assez limité
  - Des tests avec un faux résultat sont fréquents

# Intégration Continue

- Créer toutes les composants nécessaires pour un système complet
  - Construire les packages (CentOS, OpenSuSE)
  - Déployer un cloud test avec ces packages
  - Faire des tests fonctionnels dans le cloud test
- Jenkins peut gérer les dépendances entre les jobs quasi-automatiquement
- Peut stocker les artifacts dans Jenkins

# Dépendances StratusLab



# Gestion de Releases

- Faire toutes les actions pour un release final
  - Le tagging du code est faites par les jobs hudson
  - Génération des packages finals
  - Création des dépôts YUM
  - Installation et vérification du système dans une infrastructure pour la certification

# Points Forts

- Jenkins/Hudson est stable et fiable
- Simple de déployer et configurer
- Milliers des plugins pour automatiser les tâches
- Très bonne intégration avec les outils du build
- Très bonne intégration avec les outils du test
- Histoire du statut du job et une résumé des changements du code
- Facile de ajouter les nouveau jobs et éditer les jobs existants

# Pas Complètement Automatique

- Ça prend du temps pour maintenir les jobs dans Jenkins et ajouter des nouveaux
- Doit avoir quelqu'un(e) qui surveiller les résultats et forcer les gens de corriger leurs bugs
- Ça prend du temps pour créer et maintenir les machines avec le(s) bonne(s) environnement(s) pour le build et test
- L'intégration avec les ressources cloud n'est pas très utile a ce moment



# Points Négatifs

- Un serveur Jenkins pour chaque projet
  - Le contrôle d'accès et le gestion des views sont très limité
- Les configuration peut venir très complexe
- On peut facilement cacher des informations utiles dans les jobs Jenkins : p.e. les astuces pour la déploiement
- Les configurations du jobs ne sont pas versionner : difficile de distinguer si un job échoue au cause d'une changement du code ou du test
- Les tests du système prenant beaucoup de temps : les changement sont mélangés et le feedback au développeurs est retardé

# Bénéfices

- Tableau de bord pour comprendre le statut du logiciel
- Minimisation du effort pour mettre en place les processus de développement et les maintenir
- Ça rendre accessible les informations du build, test, et intégration au toutes les participants

# Résumé

- Tous les projets de développement doivent utiliser un système de l'intégration continue
  - Ça valider les changement du code rapidement
  - Permet la validation dans plusieurs environnements
  - Faciliter les discussions des problèmes/changements
- Jenkins/Hudson
  - Bon candidat pour vos projets : facile à déployer mais robuste et complet
  - Suffisamment flexible pour automatiser les test unitaires, contrôles de qualité, l'intégration des composants, et le gestion des releases

# Questions?