

# Inria's Continuous Integration portal: feedback

**Julien Nauroy**

**Inria R&D engineer**

**Laboratoire de Recherche en Informatique**

**Université Paris Sud**

**2013-10-16**

# What is Continuous Integration ?

“merging all developer working copies with a shared mainline several times a day” Wikipedia

- Comes from eXtreme Programming
- Aims at preventing the « integration hell »
- Reduces rework

Automated unit tests are closely related to C.I.

- Main practice in C.I.

# Good practices

1. Have a repository everyone can commit to
  - Any SCM: SVN, Git, Hg...
2. Commit often, at least once per day
  - Reduces risk & rework
  - start the day by updating from repository
  - smaller commits make finding bugs easy
3. Create tests ; compile & test before committing
  - commits should be working

# Continuous Integration tools

Tools can automate Continuous Integration

- E.g. Hudson & Jenkins

Need to make building and testing automatable

- makefile, ant, maven, etc

Need to access the SCM server



## Jenkins

Probably the most well-known C.I. tool

- Initial release in Feb. 2011
- Written in Java

On par with Hudson, from which it forked

- Hudson's first commit on Github : Nov. 2006
- Oracle claims to have a patent on it



Works as an independant server

- Checks out from SCM (SVN, Git...)
- Performs automated actions (build, test, ..)

# Stratuslab's Hudson instance

## Hudson

Hudson



[Personnes](#)



[Historique des constructions](#)



[Relations entre les projets](#)



[Vérifier les empreintes numériques](#)



[Dependency Graph](#)

### File d'attente des constructions

<a href="#">build_ALL_OpenSuSE</a>	
<a href="#">release_registration_OpenSuSE</a>	
<a href="#">build_storage_OpenSuSE</a>	
<a href="#">release_storage_OpenSuSE</a>	
<a href="#">build_registration_OpenSuSE</a>	
<a href="#">release_authn_OpenSuSE</a>	
<a href="#">release_client_OpenSuSE</a>	
<a href="#">build_one_OpenSuSE</a>	
<a href="#">release_distribution_OpenSuSE</a>	
<a href="#">release_image-recipes_OpenSuSE</a>	
<a href="#">release_marketplace_OpenSuSE</a>	
<a href="#">release_metadata_OpenSuSE</a>	
<a href="#">release_one_OpenSuSE</a>	
<a href="#">build_one-unpatched-3.4_OpenSUSE</a>	
<a href="#">build_one-unpatched-3.2_OpenSUSE</a>	
<a href="#">build_metadata_OpenSuSE</a>	
<a href="#">build_marketplace_OpenSuSE</a>	

All	Build_CentOS	Build_OpenSuSE	CentOS_snapshots	Certification	Install	Marketplace	OpenNebula	Req
S	W	Tâche ↓	Dernier succès		Dernier échec			
		<a href="#">authn_Release</a>	9 mo. 17 j (#27)		9 mo. 17 j (#25)			
		<a href="#">benchmarks_Release</a>	11 mo. (#14)		N/A			
		<a href="#">biocomp_update_weekly</a>	N/A		N/A			
		<a href="#">build_ALL_CentOS</a>	5 h 11 mn (#309)		N/A			
		<a href="#">build_ALL_OpenSuSE</a>	4 j 5 h (#194)		6 mo. 23 j (#78)			
		<a href="#">build_authn_CentOS</a>	5 j 5 h (#463)		3 mo. 18 j (#344)			
		<a href="#">build_authn_OpenSuSE</a>	5 j 5 h (#284)		2 mo. 13 j (#210)			
		<a href="#">build_benchmarks_CentOS</a>	5 j 5 h (#324)		2 mo. 13 j (#250)			
		<a href="#">build_benchmarks_OpenSuSE</a>	5 j 5 h (#240)		4 mo. 11 j (#125)			
		<a href="#">build_client_CentOS</a>	5 j 5 h (#700)		2 mo. 3 j (#606)			
		<a href="#">build_client_OpenSuSE</a>	5 j 5 h (#452)		17 j (#423)			
		<a href="#">build_distribution_CentOS</a>	5 j 5 h (#163)		1 mo. 23 j (#113)			
		<a href="#">build_distribution_OpenSuSE</a>	5 j 5 h (#136)		4 j 5 h (#137)			

# Inria's C.I. portal

# Overview

Inria's CI portal offers access to Jenkins instances to projects lead by Inria users

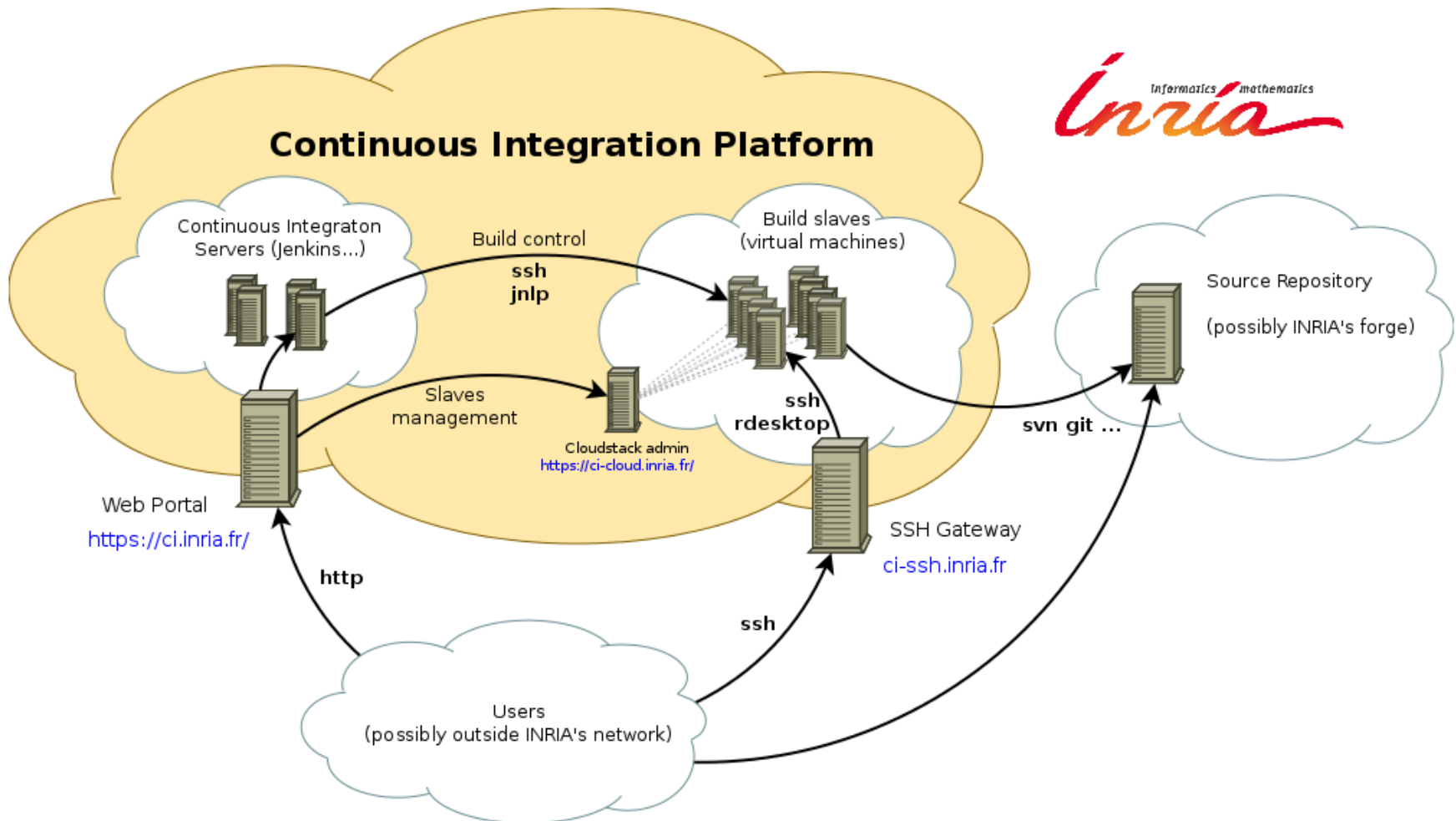
- <https://ci.inria.fr>

Basic registration is required (email@inria.fr)

One project => One Jenkins server



# How does it work ?



# User's Dashboard



Continuous Integration



Dashboard



News

Projects ▾



Users



julien.nauroy@inria.fr ▾

## Dashboard Julien Nauroy

Host your project on the Continuous Integration platform. It will provide a fully configured Jenkins server (with backup) and nodes to build your jobs. You will also be able to use your existing nodes.

Host my project

Join an existing project. Browse the project list and click the 'join' button.

Join an existing project

Gridobs\_Query

Application de requêtage du Grid Observatory

Active

Private

Jenkins

Manage project

# Creation of a project

One project = one Jenkins server

## New project

Shortname \*

pleiniere-SED

Fullname \*

Démo pour la pleinière SED

Description \*

Blah

Project visible in the project list? \*

Yes



Software \*

jenkins



Reset

Create project

# Configuration options

## Démo pour la plénière SED

Public Pending

Jenkins

Abstract

Users

Slaves

Jenkins

Logs

### Production

Installed revision : **Jenkins 1.509.2**

Jenkins

Status : **Online** [Refresh](#) [Stop](#)

### Qualification

Installed revision : **Jenkins 1.509.2 LTS**

Jenkins

Status : **Online** [Refresh](#) [Stop](#)

## Change production version

Following items will help you to manage continuous integration software versions you want to use. You will be able to update the continuous integration software version running for your project (**production**) and to test it in a sandbox (**qualification**) before pushing it on production.

To be able to fully qualify the continuous integration software version you want to push on production:

### Step 1 : Setup the Qualification Area

First, install the new continuous integration software version on the **qualification** area :  [Install](#)

### Step 2 : Synchronization

Replicate continuous integration configuration / jobs (from production) to qualification. It will allow you to test your current configuration in a sandbox.

[Production → Qualification](#)

### Step 3 : With great power comes great responsibility

Test and qualify the new continuous integration software version, and then update the production server. [✓ Push this version on production](#)

If you face problems with your newly installed continuous integration software, you can go back to the previous revision installed: [↩ Get back to previous version](#)

# Creation of a slave

## New slave pleiniere-sed

Name \*

pleiniere-sed- slave1

OK

Optional description in Jenkins

Template \*

Featured Community My Templates

- ☐ centos-6.3-i386
- ☐ centos-6.3-amd64
- ☐ ubuntu-12.04-i386
- ☒ ubuntu-12.04-amd64

Create slave

◀ Back to pleiniere-sed

**Information :** Slave's name can contain letters, digits and hyphen ('-').  
It must start with a letter and cannot end with an hyphen.

Computer offering \*

Medium Instance : 1x2.00 GHz

Small Instance : 1x1.00 GHz + 1GB RAM  
Medium Instance : 1x2.00 GHz + 2 GB RAM  
Large Instance : 2x2.00 GHz + 2GB RAM  
xLarge Instance : 2x2.00 GHz + 4GB RAM  
Minimal Windows (7 and more) Instance : 2x1.5 GHz + 2 GB RAM

Additional Disk

None

Reset

Featured templates  
(GNU/Linux) and  
Windows

# Feedback on Continuous Integration

# The projet : PHP web framework

~ 7500 lines, 4000 « of code »

Interpreted language

- Syntax errors detected at runtime

MVC architecture

Lots of refactoring to be done

- How to ensure constant quality ?

# Steps

1. Write tests
2. Automate them
3. Set up continuous integration



# A few words about software testing

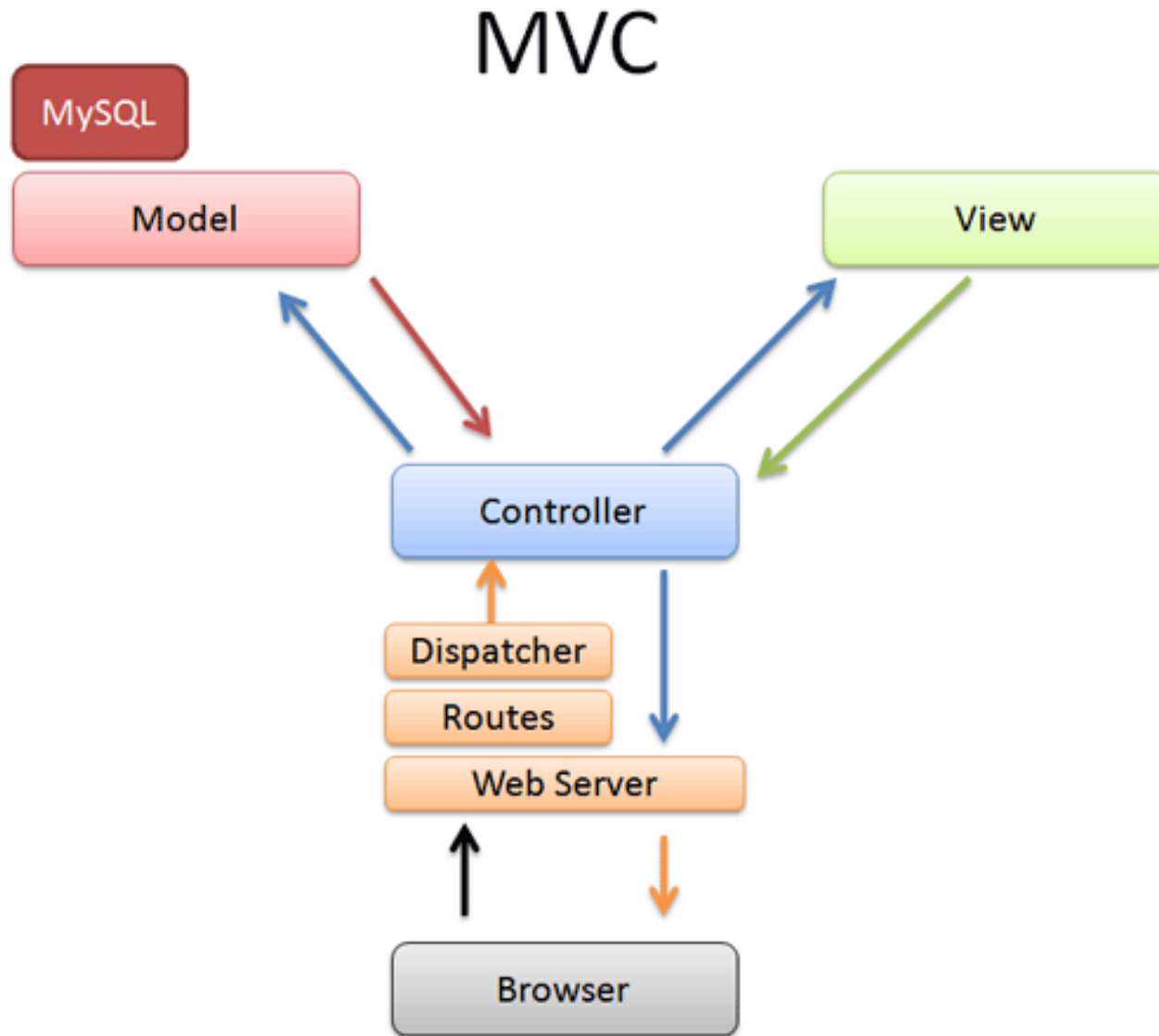
## What's software testing ?

- Partial verification of a system
- Comes in various kinds and flavors,
  - Unit tests, integration tests, performance, non-regression, robustness, vulnerability...

## Why test my framework ?

- Impact of a bug can be important
- Numerous and anonymous potential users (web)
- Related to software quality assurance (SQA)

# Specificities of testing an MVC framework



# Let's create a first unit test

```
// The email must be valid
$user = new User();
$user->login = "Julien";
$user->password = "secret";
$user->email = "invalid";
$returnCode = $user->save();
$this->assertEquals(
    User::EMAIL_INVALID,
    $returnCode);
```

# Kind of easy, isn't it ?

Each « unit » has to be tested

- Classes, methods...

*A de facto standard* : xUnit (here, PHPUnit)

- Very easy to get a hold on
- Many tutorials available

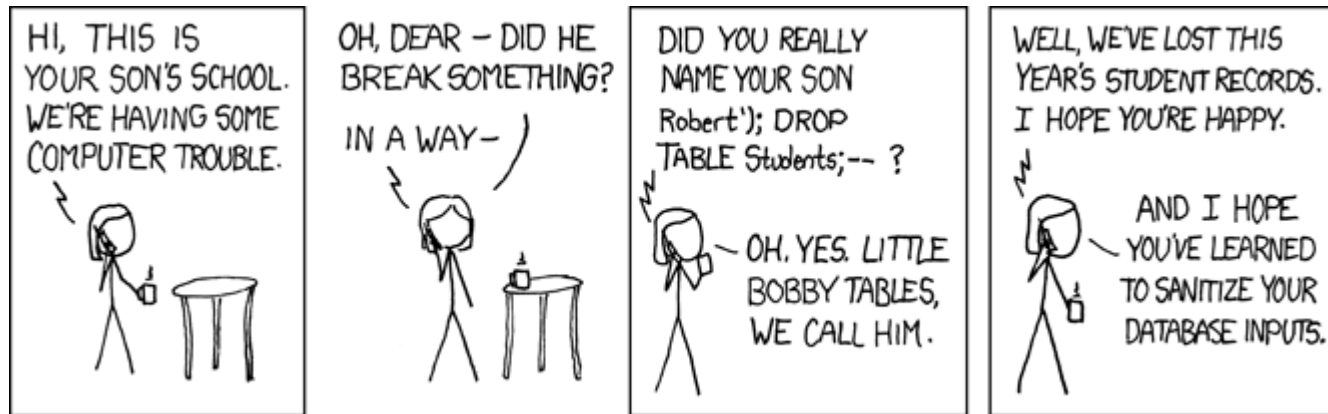
Use simple assertions

- assertTrue, assertFalse
- assertEquals
- assertCount
- assertContains

# Doing it well is a bit hard

Try to think about everything

- Create a new user
  - User's name is valid (policy)
  - User's name is available
  - User's email is valid and available
  - No SQL syntax in the name (SQL injection)
  - No HTML syntax (XSS)

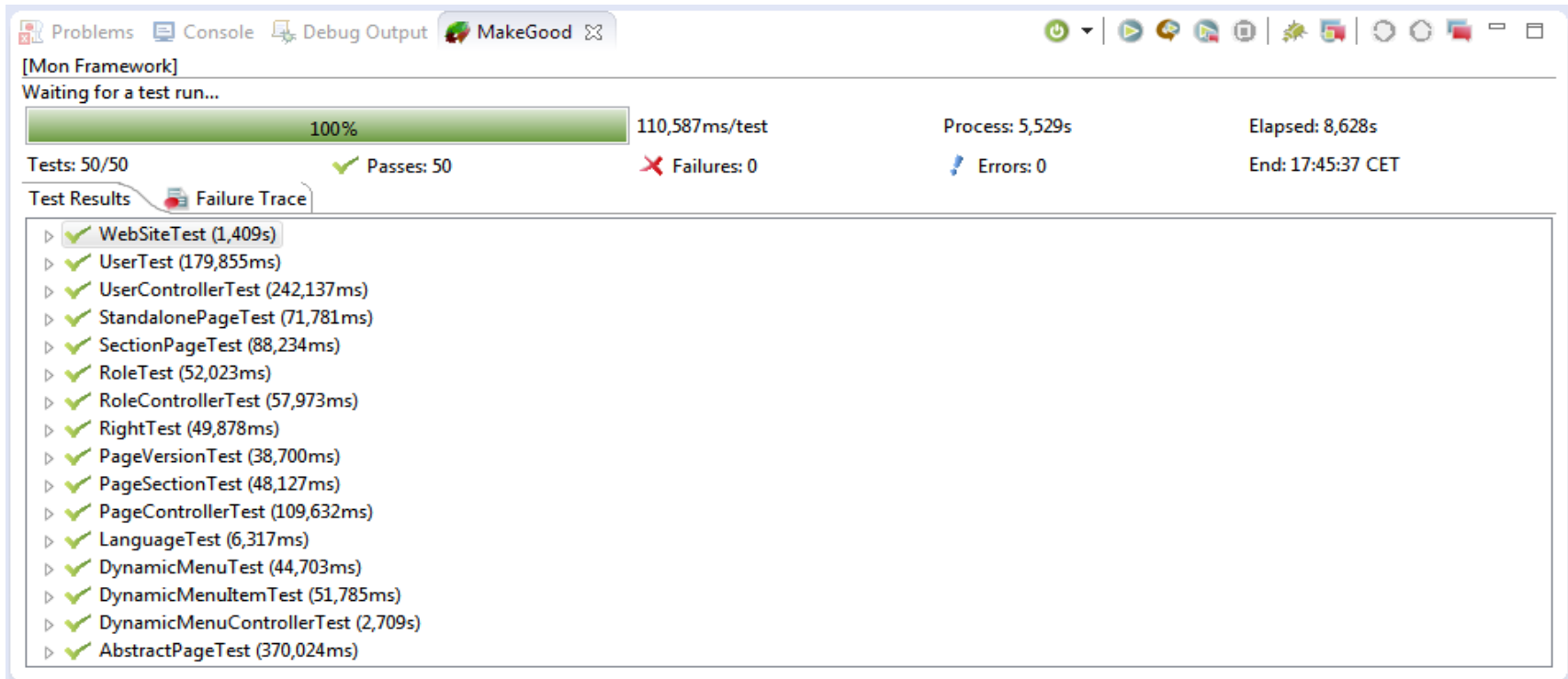


# After some elbow grease...

~1000 lines of code for unit tests

A few days of development

50 functions, 567 assertions



The screenshot shows the Visual Studio Test Results window for a project named 'Mon Framework'. The window displays a progress bar at 100% completion, indicating that all tests passed. The summary shows 50 tests, 50 passes, 0 failures, and 0 errors. The total elapsed time is 8,628 seconds. The test results list includes:

- ✔ WebsiteTest (1,409s)
- ✔ UserTest (179,855ms)
- ✔ UserControllerTest (242,137ms)
- ✔ StandalonePageTest (71,781ms)
- ✔ SectionPageTest (88,234ms)
- ✔ RoleTest (52,023ms)
- ✔ RoleControllerTest (57,973ms)
- ✔ RightTest (49,878ms)
- ✔ PageVersionTest (38,700ms)
- ✔ PageSectionTest (48,127ms)
- ✔ PageControllerTest (109,632ms)
- ✔ LanguageTest (6,317ms)
- ✔ DynamicMenuTest (44,703ms)
- ✔ DynamicMenuItemTest (51,785ms)
- ✔ DynamicMenuControllerTest (2,709s)
- ✔ AbstractPageTest (370,024ms)

# Bonus

Test + debugger traces = Code coverage

Code Coverage:  3,45 %

Code Coverage:  88,89 %

```
// Create the linkURL variable depending on the type of item
switch($this->dynamicMenuItemType)
{
```

```
    case 'genericLink':
        $this->linkURL = $this->dynamicMenuItemValue;
        break;
    case 'sectionIDLink':
        $this->linkURL = Controller::makeRoute (
            'page/viewSection/' . $this->dynamicMenuItemValue
        );
        break;
    case 'pageIDLink':
        $this->linkURL = Controller::makeRoute (
            'page/view/' . $this->dynamicMenuItemValue
        );
        break;
    case 'container':
        $this->linkURL = '';
        if ($fetchSubItems) {
```

# Automation

Tests must be automated to be used as C.I.

- Create makefile, use Ant, Maven, ...

For this project, use of Phing (Ant for PHP)

Not an easy task !

- Need lots of tools like pear, PHPUnit, Xdebug, ...
- Lots of trial and error

Need to create output files to pass to C.I. tool



# build.xml for Phing

```
<target name="tests" depends="prepare">
    <coverage-setup database="reports/coverage.db">
        <fileset refid="unitTests" />
    </coverage-setup>
    <phpunit haltonfailure="true" printsummary="true" codecoverage="true">
        <formatter type="xml" todir="reports" outfile="testsuites.xml" />
        <formatter type="clover" todir="reports" outfile="coverage.xml" />
        <batchtest>
            <fileset refid="unitTests" />
        </batchtest>
    </phpunit>
</target>

<target name="phploc">
    <phploc reportType="csv" reportName="phploc" reportDirectory="reports/">
        <fileset refid="WebCodeBaseCoreFiles" />
    </phploc>
</target>
```

# After some more work...

```
> phing tests
Buildfile: build.xml
```

```
XXX > prepare:
```

```
    [delete] Deleting directory reports
    [mkdir] Created dir: reports
```

```
XXX > tests:
```

```
[coverage-setup] Setting up coverage database for 16 files
[phpunit] Total tests run: 50, Failures: 0, Errors: 0,
           Incomplete: 0, Skipped: 0, Time elapsed: 14.44883 s
```

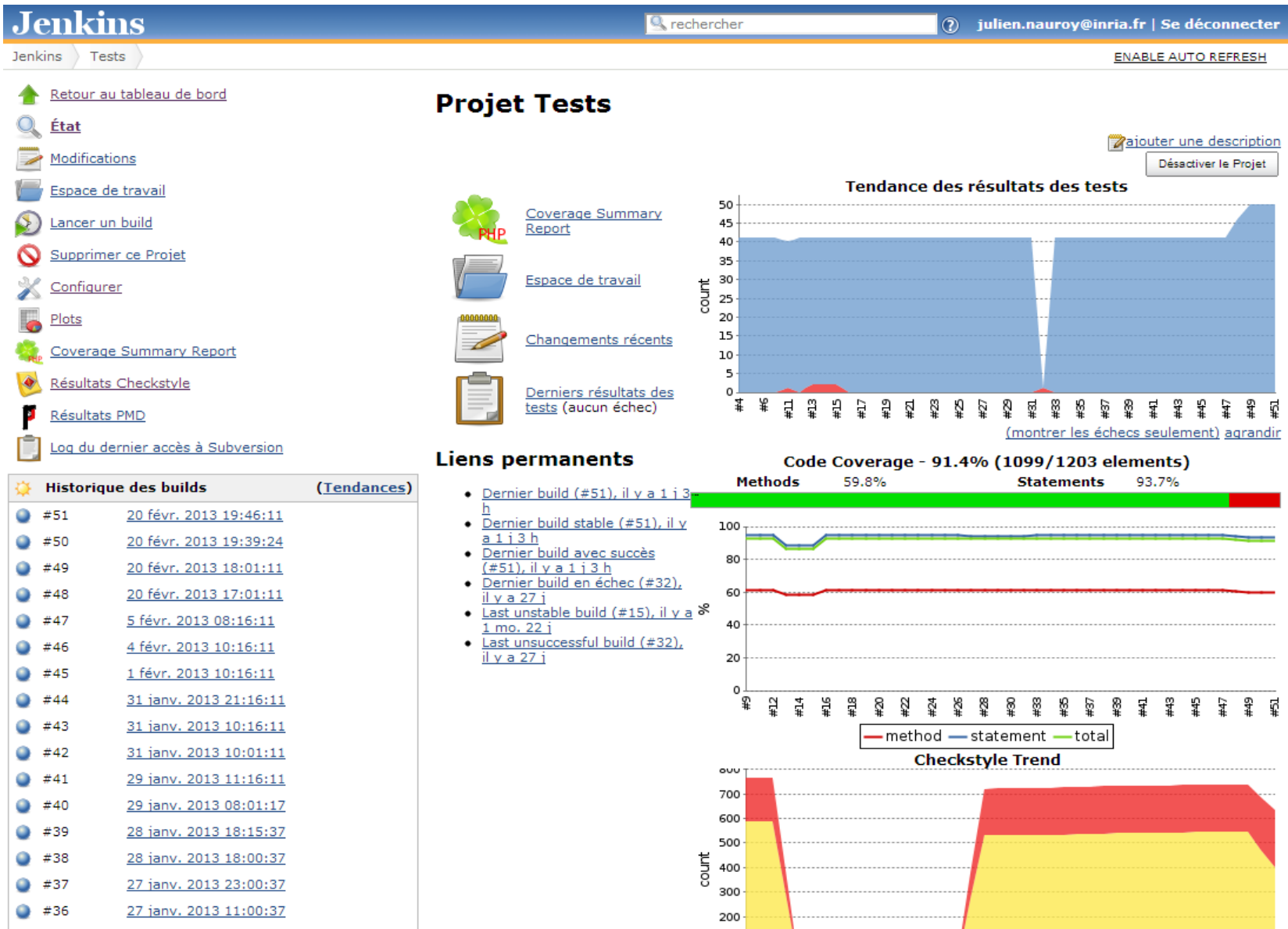
```
BUILD FINISHED
```

```
Total time: 17.2220 seconds
```

# Integration within Inria's C.I. portal

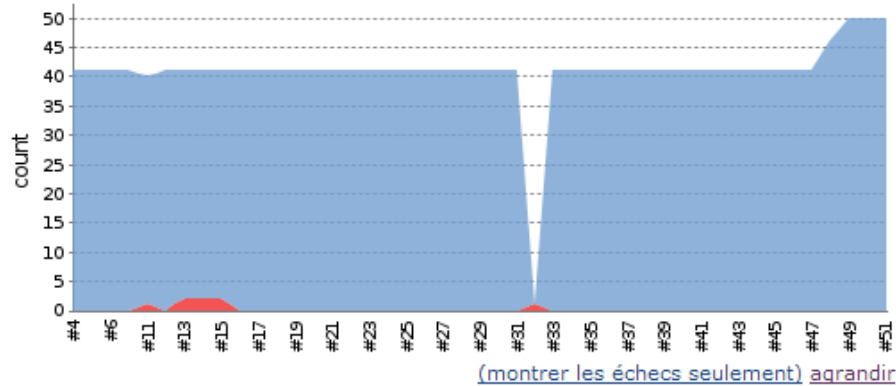
1. Create a project and slaves
  - slaves are VMs, must be configured (takes time)
2. Create a Job or Project
  - Names are confusing, at least in french
3. Configure the Job
  - May be painful & time consuming
4. Repeat until happy
  - Multiple jobs can be created for different actions

# Results

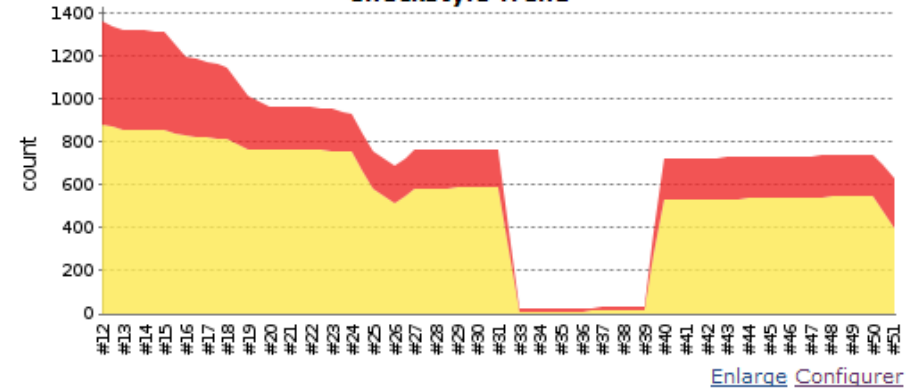


# Results (2)

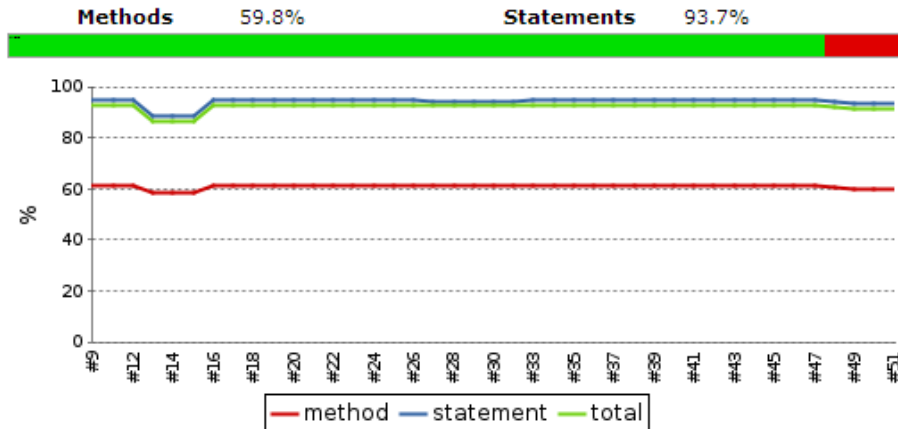
Tendance des résultats des tests



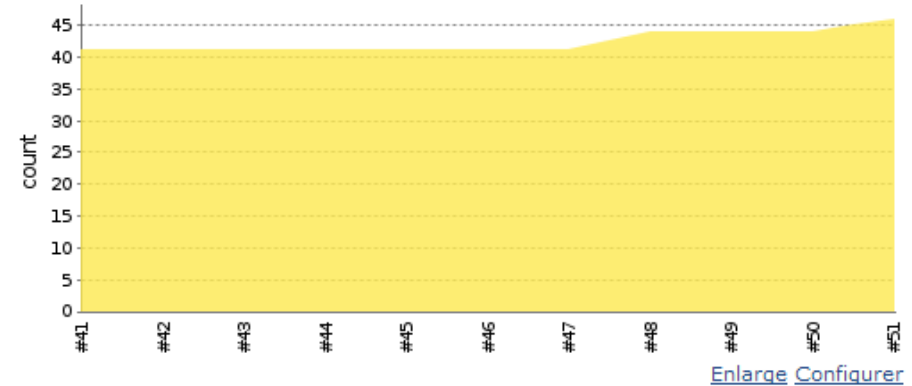
Checkstyle Trend



Code Coverage - 91.4% (1099/1203 elements)



PMD Trend



# Summary

# Summary

C.I. in itself is not very hard to set up

- At least for a small project

Part of any SQA policy

- Unit tests are now a standard way to ensure reliability
- Do you want to build quality software ? You'll need it !

Has some other advantages

- Nightly snapshots, generation of documentation, ...
- Quality metrics, encourages the « boy scout rule »

# To go further

Emmanuel Jeanvoine's upcoming presentation of Inria's C.I. portal

- November 8th, 2013

Upcoming Hands-on on test & C.I.

- Date not decided yet
- Probably in english
- Java + Maven + Jenkins



# Quality Assurance: ISO/CEI 9126

## Functionality

- Suitability
- Accuracy
- Interoperability
- Security

## Reliability

- Maturity
- Fault Tolerance
- Recoverability

## Usability

- Understandability
- Learnability
- Operability
- Attractiveness

## Efficiency

- Time Behaviour
- Resource Utilization

## Maintainability

- Analyzability
- Changeability
- Stability
- Testability

## Portability

- Adaptability
- Installability
- Co-Existence
- Replaceability