



# Intégration continue

# Intégration quoi ?

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel. Elles consistent à **vérifier automatiquement à chaque modification de code source que le résultat des modifications ne produit pas de régression** de l'application en cours de développement.

— [Wikipedia](#)

# En français ?

T'as le droit de tout péter, mais t'es obligé de **mettre les autres au courant**.

— @n1k0

# Le but de l'intégration continue

- Avoir **confiance** en son propre code et celui de l'équipe
- Renforcer la qualité du code en **responsabilisant** l'équipe
- Déployer plus **sereinement**
- Soulager les perfectionnistes pressés et sensibles au **stress**
- Prendre l'**apéro** plus tôt

# Comment ?

## Régulièrement et automatiquement :

- Contrôler l'**intégrité** unitaire et fonctionnelle
- S'assurer de l'absence de **régressions**
- Surveiller le respect des **conventions** de codage
- Mesurer la **couverture** du code
- Signaler les **erreurs** de codage
- Générer des **rapports** synthétiques utiles

# Présentations

- Laurent/Cal : Jenkins
- Christian A. : l'approche Nuxeo
- Sébastien : Travis-CI
- Oleg : Sonar
- Éric Legay (CSNSM) : une ressource IN2P3 ?

# Jenkins

## Deploiement et Utilisation

- StratusLab (<https://hudson.stratuslab.eu/>)
  - « Continuous Testing »: Exécuter les tests unitaires sur plusieurs OS après chaque commit
  - « Continuous Integration »: Construire les packages, déployer un système et faire les tests fonctionnels
  - « Release Management »: Générer les packages, certifier le release, et créer les dépôts YUM
- PSPA (134.158.75.115:8080)
  - Tests continus : Compilation et exécution sur une machine cible SL6 après chaque commit
  - Déploiement d'une application web

**Jenkins**





# Jenkins

•PSPA

Jenkins  
PSPA-Dev [Jenkins]

Jenkins ENABLE AUTO [ajouter une d](#)

[Nouveau Job](#)  
[Utilisateurs](#)  
[Historique des constructions](#)  
[Administrer Jenkins](#)

**File d'attente des constructions**  
File d'attente des constructions vide

**État du lanceur de constructions**

#	Statut
1	En attente
2	En attente

S	W	Name	Dernier Succès	Dernier Echec	Dernière Durée
		<a href="#">Geant</a>	7 j 5 h (#30)	1 j 1 h (#24)	1 mn 38 s
		<a href="#">PSPA-Dev</a>	5 j 5 h (#73)	4 mn 42 s (#106)	1 mn 57 s
		<a href="#">PSPA-stable</a>	N/A	1 h 17 mn (#31)	15 mn

Icône: [S](#) [M](#) [L](#)

[Légende](#) [RSS pour toutes](#) [RSS pour tous les échecs](#) [RSS juste pour les derniers](#)

# Jenkins

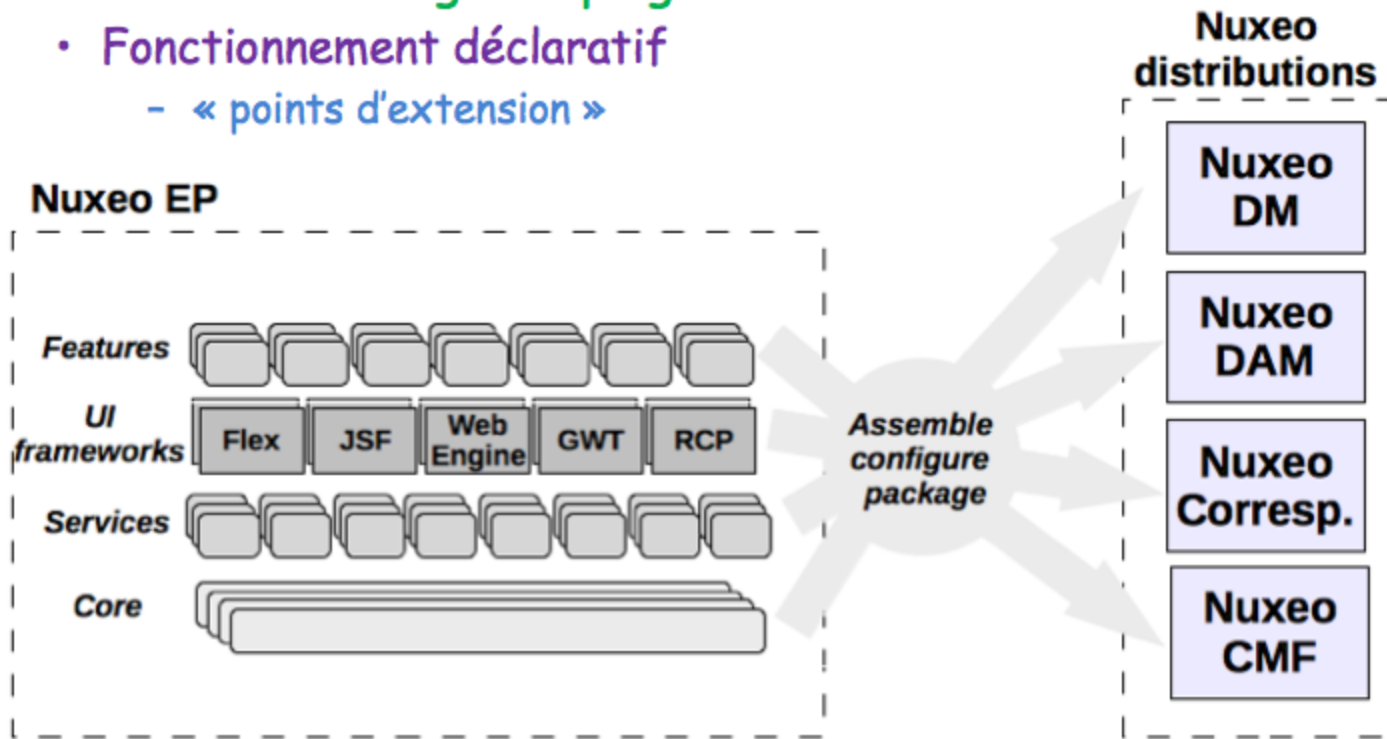
## Avantages et Inconvénients

- Utilisation simple vers une interface web
  - Bonne intégration avec les SCMs
  - Gestion des plusieurs OS simultanément
  - Dépendances possible entre tâches
- Pas d'historique des configuration de test
  - Les configurations ne sont pas sauvegardées

# Approche Nuxeo

# Développer avec Nuxeo

- Le logiciel Nuxeo
  - Open source
  - Architecture client-server
  - C'est un assemblage de plugins Java/XML
    - Fonctionnement déclaratif
      - « points d'extension »



# Approche Nuxeo

# Développer avec Nuxeo

---

- Eclipse
  - Nuxeo a fourni un plugin « nuxeo »
  - Plus un Wizard pour construire des Nuxeo Artifact
    - Operation
    - Component
    - Document adapter
    - Document Listener
    - Automation gadget
    - Action/Controller/Service Bean
    - Security
  - Lancement et debugging du server à travers Eclipse
  - Déploiement des artifacts développés
  - Construction d'un plugin Nuxeo (construction, configuration, packaging)
    - Un plugin Nuxeo peut être déployé à partir de l'Admin Center du client Nuxeo (donc sans visibilité directe du code du serveur)

# Approche Nuxeo



# Développer avec Nuxeo

---

- Maven
  - Est un autre moyen de configurer/packager/déployer des plugin nuxeo
  - L'environnement Eclipse sait configurer le *pom.xml*/correspondant
- Junit
  - Le plugin Nuxeo pour Eclipse fournit aussi un plugin Junit spécialisé pour Nuxeo.
    - Templates de tests pour chaque type d'artefact
    - Possibilité de configurer un environnement Jenkins pour automatiser les tests ainsi définis
- Nuxeo Shell
  - Client mode line qui permet de naviguer dans la base des plugins installés
  - Permet d'activer manuellement les opérations, les tests
  - Scripting pour lancer des batches

# Travis-CI

*A hosted continuous integration service for the open source community.*

- <https://travis-ci.org/>
  - MIT license
  - written in Ruby (nobody's perfect)
  - leverages vagrant for VM provisioning
  - **integrated** w/ GitHub
  - first class support for:
    - ▶ C, C++, Clojure, Erlang, **Go**, Groovy, Haskell,
    - ▶ Java, JavaScript (with Node.js), Perl, PHP, Python, Ruby, Scala
  - VM: ubuntu-32b-12.04
  - configuration thru a **YAML** file: `.travis.yml`
  - a **Go** example: `go-ffi`
-

# Travis-CI

```
language: go
```

```
before_install:
```

- sudo apt-get update -qq
- sudo apt-get install libffi-dev -qq

```
install: go get -d -v ./pkg/... && go build -v ./pkg/...
```

```
script: go test ./pkg/...
```

```
notifications:
```

```
  email:
```

```
    recipients:
```

- binet@cern.ch

```
    on_success: change
```

```
    on_failure: always
```

# Sonar

## Objectifs

- ❑ améliorer la lisibilité du code
- ❑ se conformer aux standards d'écriture
- ❑ vérifier le niveau de documentation in-ligne

Dissémination

Debugging

- ❑ détecter les duplications de code
- ❑ implémenter les tests unitaires
- ❑ contrôler le taux de couverture
- ❑ détecter les codes «morts»

- ❑ et toujours : le gestion des versions

Versionning

# Sonar

## Chaîne d'outils

Trop (de docs, d'outils...) peut être aussi néfaste que pas assez

Dissémination

- Eclipse (<http://www.eclipse.org>)
- Sonar (<http://www.sonarsource.org>)

Debugging

- JUnit (<http://junit.sourceforge.net>)

versionning

- svn (<http://subversion.tigris.org>)

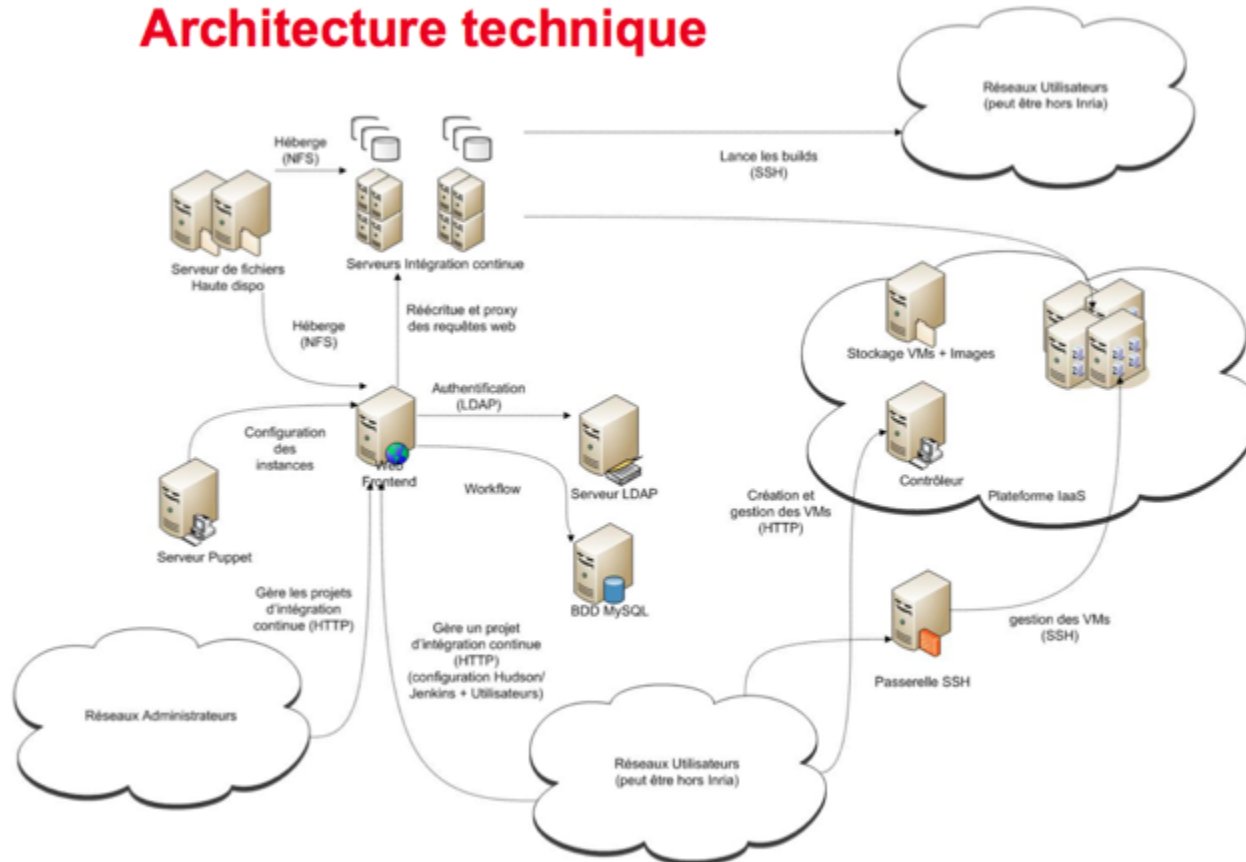
# Une ressource IN2P3 ?



**Plateforme nationale  
d'intégration continue**

# Une ressource IN2P3 ?

## Architecture technique



# 2e journée *LoOPS*

- LoOPS: développements logiciels Pour la Science (ou Paris Saclay)
- 11 décembre dans le "Grand amphi" du bâtiment 640 de l'Université Paris-Sud, sur le plateau de Saclay.

Cette journée sera axée sur le test et le build. Elle mêlera des retours d'expériences, des ateliers et des présentations de pratiques et d'outils.

	<b>9h00</b>	Accueil	
Exposés pléniers	<b>9h15</b>	Les tests, comment les aborder ? (Frédéric Voisin, équipe ForTesSE, LRI)	
	<b>10h00</b>	Retour d'expérience avec Sonar (Oleg Lodygensky, LAL)	
	<b>10h30</b>	Pause	
	<b>11h00</b>	Jenkins, votre serviteur (Charles Loomis, LAL)	
	<b>11h45</b>	N'ayez pas peur d'OCaml (Maxence Guesdon, INRIA Saclay)	
	<b>12h00</b>	Utiliser Cmake pour la reconstruction, les tests, le packaging (Alexandre Abadie, INRIA Saclay)	
	<b>12h20</b>	Déjeuner	
Ateliers parallèles	<b>13h30</b>	Atelier Cmake	Atelier OCaml
	<b>15h45</b>	Pause	
	<b>16h00</b>	Table ronde: retours sur la journée, questions, JDEV2013, suite du réseau	
	<b>17h00</b>	Fin de la journée	



**Voilà !**