270 Boulevard Georges Clemenceau 59700
Marcq-en-Barœul, France

Rapport de stage d'ingénieur

Création d'une prototype pour gérer images satellitaires sur le cloud

Élève : **Pere Cortés Manyanic**

Diplôme: **Ingénieur**

Directeur de stage en entreprise: **Sébastien Robin**

Correspondant de stage à TELECOM ParisTech : **Maurice Gagnaire**

Dates du stage : **2-07-2012 au 21-12-2012**

Avant-propos

Le présent rapport vise à présenter mon stage de fin d'études au sein de l’entreprise Nexedi, entre le 2 juillet et le 21 décembre 2012, dans les domaines du développement web, et en particulier, la réalisation d’une prototype pour gérer images satellitaires sur le cloud.

Mon but ne sera pas d’analyser en profondeur les détails du code qui a été écrit, toutefois une petite partie est ajouté aux annexes, sinon d'expliquer de forme global le travail fait pendant le stage.

Avant toute chose, je tiens à remercier Sébastien Robin pour m’avoir accepté au sein de l’entreprise Nexedi dont il est chef de projets. Aussi, je veux lui remercier pour moi donner l'opportunité de travailler avec lui pendant cette sis mois.

Ainsi comme, je remercie à toute l’équipe de Nexedi pour son accueil laquelle a permis de trouver mon place rapidement. Vraiment, ils sont comme une petite famille.

Aussi, Je remercie Maurice Gagnaire qui a accepté d'être mon tuteur école et qui m'a aidé beaucoup depuis mon arrivé à Télécom-Paristech.

Je veux aussi mettre l'accent sur les valeurs morales défendues par Nexedi, son rôle dans le développement et la défense des intérêts du monde du libre. Ils m'ont montré un autre point de vue sur les technologies numériques.

Sommaire

Introduction

L’objectif principal de mon stage était démontrer qui est possible d'afficher images larges depuis tablettes ou smartphones.

Bien que nombreux inconvénients ont apparu, le premier prototype a été fait après 3 mois de travail. Toutefois, il n’était pas fini complètement.

Alors, mon chef de stage m'a proposé améliorer le système de test de l’entreprise. C'est pour ça que mon rapport est divisé en deux blocs : le projet SafeImage et l'amélioration de erp5 test node.

Il faut dire que les deux projets sont complémentaires car au fin de mon stage, j'ai ajouté quelques tests pour le projet SafeImage en utilisant les améliorations que j'ai déjà fait sur erp5testnode.

Le projet SafeImage n'est pas encore utilisé dans le sien de Nexedi sinon ils sont en train de l'ajouter sur autres projets déjà existants. Par contre, les améliorations de erp5testnode ont été utilisé depuis le premier jour en faisant plus dure le processus de développement puisque un mauvais mis à jour du logiciel pouvais laisser sans système de test aux développeurs de l'entreprise.

I. Nexedi

1. Présentation de l’entreprise

Créée en 2001 par Jean-Paul Smets, Nexedi est une société internationale implantée en France, en Allemagne, aux États-Unis, au Brésil, au Japon, et au Sénégal, qui propose aux entreprises et aux gouvernements des services de conseil, de développement et d’assistance 24h/24 pour leur permettre de faire évoluer leur système d’informations vers des solutions libres.

L’entreprise est à l’origine de ERP5, logiciel de gestion intégré en licence libre. Ce logiciel, basé sur le serveur d’application Zope, a été récompensé comme meilleur projet ERP en 2004. ERP5 est actuellement utilisé par des organisations de toutes tailles et le service d’assistance continue fourni par Nexedi aux entreprises utilisant son logiciel représente aujourd’hui la majeure partie des revenus de l’entreprise. Les principaux clients œuvrent dans les domaines de l’aérospatial, de la finance, du transport, ou encore dans le secteur publique.

Cependant, Nexedi est aussi un acteur reconnu de la recherche informatique, en particulier dans le monde du libre. L’entreprise est membre de System@tic Innovation Cluster, Foundation for a Free Information Infrastructure (FFII), Free Cloud Alliance, OW2 Consortium, et l’Association Francophone des Utilisateurs de Logiciels Libre (AFUL).

2. Les projets de Nexedi

Nexedi propose de nombreux services aux entreprises et de nombreuses applications et logiciels open source. Son principal objectif est la liberté de la information et le Cloud Computing. Voici les principaux projets développés par Nexedi:

ERP5: logiciel de gestion intégré libre, il est le produit plus important de l'entreprise. ERP5 est basé sur une base de données Zope et est codé en Python. Il est l'ERP libre le plus célèbre et performant.

UNG Docs: application web permettant de créer, d'éditer et de partager des documents. Sous sa forme 1.0, cette application est codée en Python et repose sur ERP5. Elle a pour vocation à devenir une application JavaScript libre proposant un stockage en Cloud réparti. Il est le Google Docs livre.

TioLive: ensemble de solutions applicatives disponibles en ligne en tant que SaaS (Software as a Service). TioLive propose un ERP, un logiciel de gestion de la relation client (CRM), une solution de gestion électronique de documents (GED), un service de messagerie, un chat et un service VoIP (Voice over IP).

SlapOS: système libre de Cloud Computing réparti. SlapOS se remarque par sa haute compatibilité avec les différents systèmes d'exploitation et la plupart des environnements de programmation. Il est capable de gérer tout aussi bien des bases de données relationnelles (MySQL, MariaDB), NoSQL (KumoFS, Memcached, ZEO), ou du stockage de blocs (Sheepdog, nbd). SlapOS est basé sur le démon SlapGrid qui permet de gérer l'installation de logiciels, ainsi que la création d'instance et leur destruction, selon le concept de la grille informatique.

ViFiB : fournisseur d'accès internet français, ViFiB rembourse la connexion haut débit par fibre optique aux clients qui acceptent d'héberger chez eux deux ordinateurs alimentés en électricité et connectés à internet par IPv6 .

II. Le projet SafeImage

1. Introduction

SafeImage est une application qui a comme objectif démontrer que c'est possible d'afficher grandes images depuis le navigateur des dispositifs qui n'ont pas beaucoup de puissance, par exemple les smartphones.

SafeImage est un projet de Nexedi, concrètement il fait partie du projet TSXX qui utilise le logiciel libre ERP5 pour gérer les images prises pour les satellites de l'ASE (Agence spatiale européenne). Ainsi comme, la gestion de ventes aux clients est inclue aussi.

Parfois, les clients ont besoin de voire les images depuis son smartphone ou tablette et alors, la problématique de la manque des ressources est soulevé car la taille moyenne d'une image satellitaire est d'environ 1Gb et ces types de dispositifs ne sont pas capable de les afficher. Quand même, les dispositifs plus puissants offrent des résultats très moindres puisque il s'agit de télécharger la totalité de l'image chaque fois, quand peut-être ils veulent seulement vérifier si l'image a été prise correctement.

SafeImage face à ce problème en coupant les images en différents « tiles ». Cette tache est fait pour Zoomify, une librairie qui sera expliqué plus tard.

Grâce à la division de la image, le client peut seulement télécharger les morceaux des images qu'il désire. Ainsi beaucoup de ressources sont économisé et le plus important : la performance améliore.

Pour résumer, les clients peuvent seulement voire les morceaux de la image qu'ils veulent sans besoin de télécharger toute l'image. Par ailleurs, ils peuvent en faire partout où ils veulent.

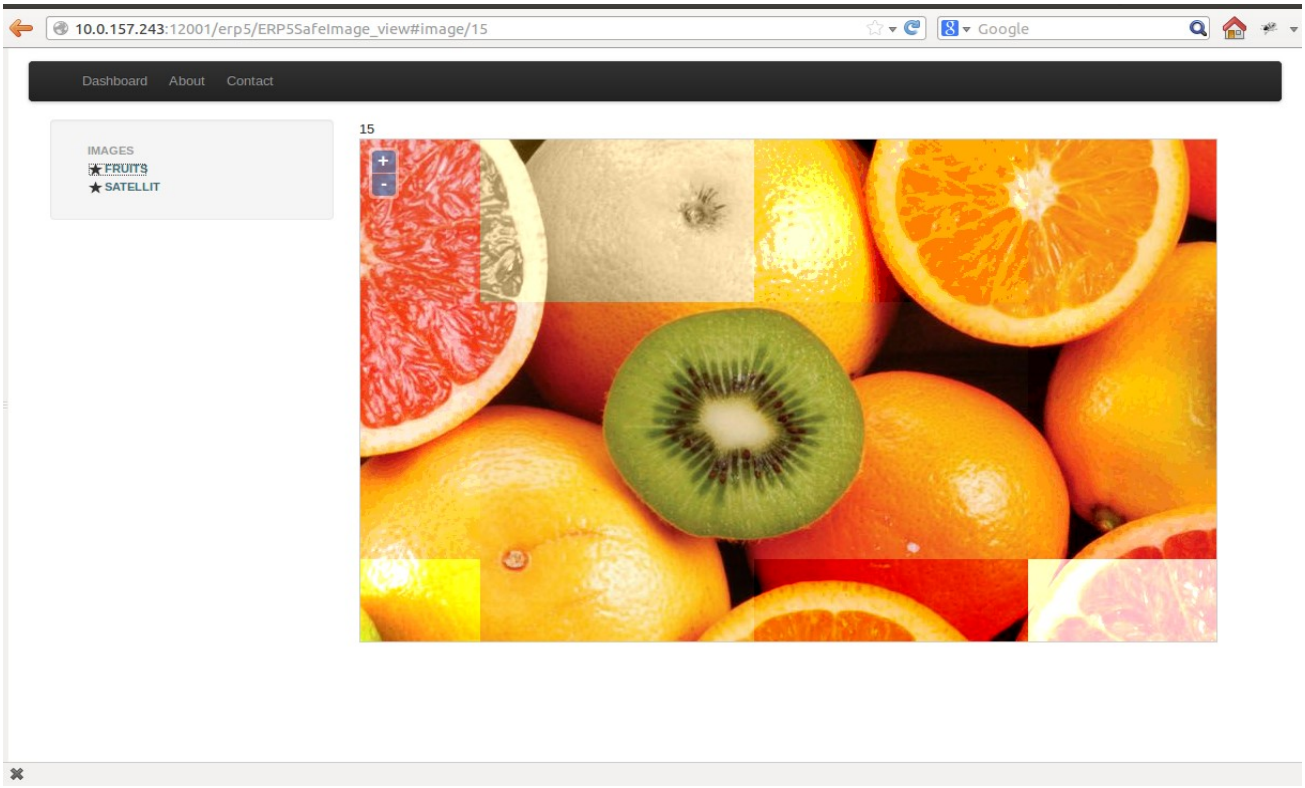
La pierre angulaire de SafeImage est ERP5. Grâce à ERP5, SafeImage a une très bonne sécurité, une couplage faible, flexibilité et l'opportunité d'ajouter nouveaux fonctionnalités.

En dehors de ERP5, SafeImage utilise autres logiciels et librairies aussi. Concrètement :

- **OpenLayers :** Librairie JavaScript livre qui permet de charger, afficher et rendre les images. OpenLayers est utilisé sur le côté-client pour afficher les « tiles » à travers des « canvas », la nouveau fonctionnalité fournit pour HTML5 qui permet d'afficher et manipuler images plus facilement. Aussi, OpenLayers fournisse autres fonctionnalités pour faciliter l'interaction à niveau d'utilisateur. Malheureusement, la version 2.0 de OpenLayers n'est

pas encore compatible avec le « canvas », alors une petite « patch » a été ajouté pour adapter OpenLayers dans SafeImage. Néanmoins, la nouvelle version, qui supportera OpenLayers, sera publiée tôt.

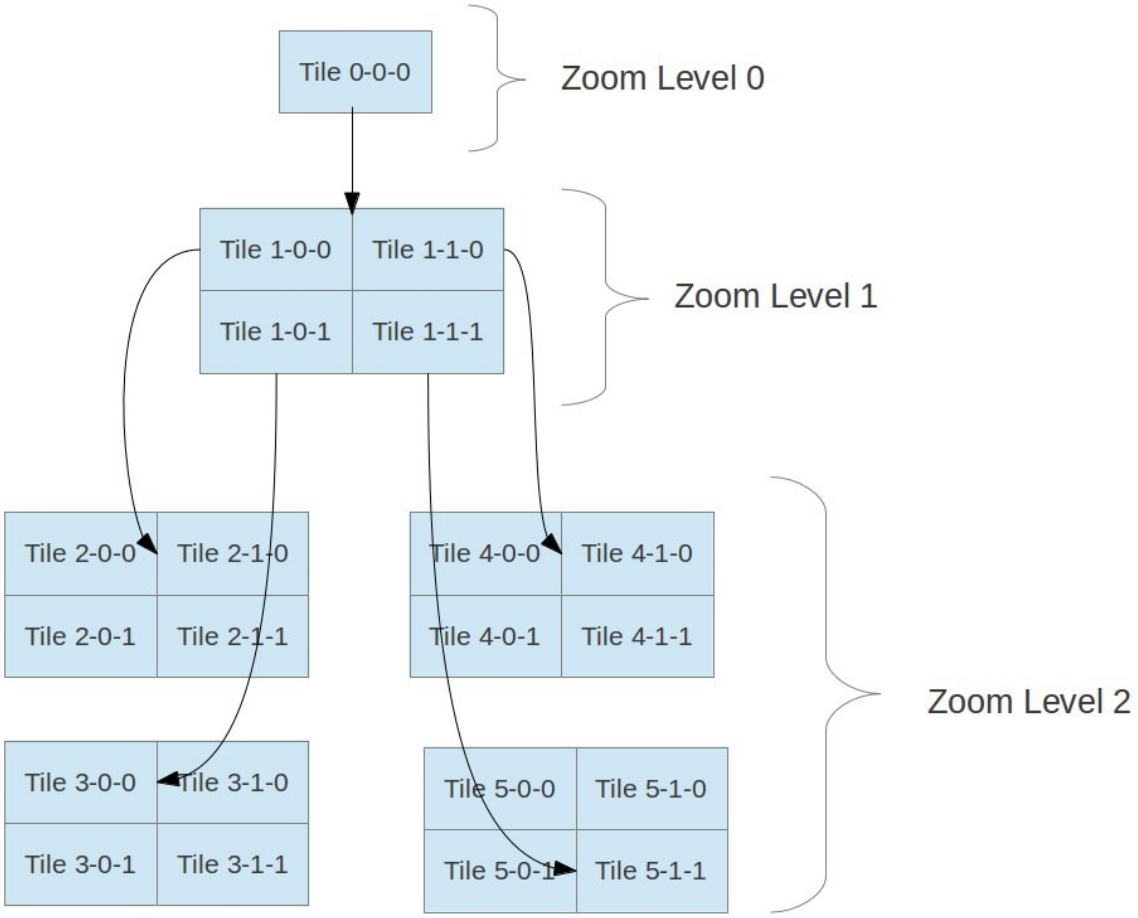
- **Pixastic :** Librairie JavaScript livre qui gère, manipule et applique les algorithmes aux images. Sur prototype de SafeImage présenté, plusieurs algorithmes ont été implémenté pour montrer les avantages d'utiliser Pixastic. Évidemment, selon les besoins de l'application les algorithmes auraient d'être changer ou modifier.
- **jQuery :** La célèbre librairie JavaScript, qui permet de manipuler la DOM facilement. Elle est utilisé pour faire les requêtes AJAX au serveur et pour définir quelques fonctions utiles.
- **IcanHaz :** Librairie JavaScript pour définir et créer ton propre template à côte-client. Sur SafeImage, un template est défini pour afficher les « tiles » de l'image depuis le navigateur.
- **Zoomify :** Librairie compatible avec plusieurs langages qui permet de couper les images en tiles. Sur le cas de SafeImage, le version Python est utilisé.



L'image ci-dessus montre l'apparence de SafeImage depuis une navigateur. L'utilisateur a au côté gauche de la fenêtre, la liste des Images hébergés dans ERP5. Après, l'utilisateur chose l'image qu'il veut et avec OpenLayers, il peut glisser sur le canvas pour afficher les morceaux désirés. Ainsi comme, il peut zoomer l'image.

2. Zoomify

Avant de rentrer dans les détails, il s'agit de comprendre mieux comme le processus de coupage est fait. L'image suivante montre graphiquement comme Zoomify fonctionne.



D'abord de tout, Zoomify calcule les niveaux de zooms. Pour faire ça, la taille de l'image est divisé pour la taille de la « tile » jusqu'à le résultat est plus petit que zéro. Chaque fois que la division est plus grand qu'une, une nouveau niveau de zoom est ajouté. Après , Zoomify gère chaque niveau de zoom indépendamment. Par exemple : La niveau zoom 0 est toujours l'image original réduite pour l'image du tile, dans SafeImage est 256 x 256 pixels. Les autres niveaux dépendent de la taille de l'image originale.

Par exemple Zoomify coupe un image qui a 1600 px de largeur et 1200 de hauteur suivi le procédure suivant:

D'abord de tout, Zoomify divise la taille original entre deux jusqu'au résultat est plus petit que la taille minimum du tile. Sur ce cas est 256. Donc, le résultat est : scale_vector = [(400,300) , (800,600) , (1600,1200)]. Chaque scale_vector est un niveau de zoom différent. Donc, il y a trois niveaux de zoom plus l'image original mis à l'échelle par la taille du « tile ».

Alors en commençant pour le plus petit. Chaque scale_vector est divisé pour la taille du « tile », la parte entière du résultat est prise et ensuite le module est fait pour ajouter un « tile » extra si il y a besoin. Les calcules sont les suivantes :

zoom level = (400,300)

- rows = [400/256] = 1 → 400 % 256 > 1 → rows = 2
- columns = [300/256] = 1 → 300 % 256 > 1 → columns = 2
- partial_tiles = 2* 2 = 4

zoom level = (800,600)

- rows = [800/256] = 3 → 800 % 256 > 1 → rows = 4
- columns = [600/256] = 2 → 600 % 256 > 1 → columns = 3
- partial_tiles = 4 + 4*3 = 16

zoom level = (1600,1200)

- rows = [1600/256] = 6 → 1600 % 256 > 1 → rows = 7
- columns = [1200/256] = 4 → 1200 % 256 > 1 → columns = 5
- partial_tiles = 16 + 7*5 = 51

nombre de tiles total = 51 + 1 (image original) = **52**

Quand tous les tiles sont créés. Zoomify crée une fichier XML, qui contiens le numéro des tiles, la largeur et la hauteur de l'image originale. Cette fichier est utilisé pour OpenLayers pour rendre et afficher les tiles correctement.

3. Fonctionnement de SafeImage

Avant mentionné, ERP5 est la pierre angulaire de SafeImage, qui finalise avec la création du business template erp_safeimage, avec ses éléments respectifs. Globalement erp5_safeimage contient :

- Portal types : Image Tile Transformed et Image Tile Group
- Fonctions externes : Elles sont utilisées pour implémenter Zoomify et Selenium Test dans SafeImage.
- Scripts Python : Trois scripts qui sont utilisé pour l'échange d'information entre le serveur et le client à travers de les requêtes AJAX.
- Portal Skins : Dossier qui contient tous les fichiers relatifs à SafeImage.
- Unit Test : testSafeImage, qui teste le fonctionnement correct de SafeImage dans ERP5.
- Fichiers de côté-client : Tous les fichiers JavaScript et HTML nécessaires pour rendre, processeur et afficher les images hébergées dans ERP5.
- Test fonctionnelle : safeimage_zuite, avec deux méthodes externes de Selenium, permet de vérifier le fonctionnement correct de SafeImage depuis le point de vue de l'utilisateur. Principalement, il test l'interaction de l'utilisateur avec OpenLayers.

Dans ERP5, tous les objets doivent avoir au moins un portal type associé. Le portal type est un conteneur pour l'objet car il défini la classe qui est invoqué quand l'objet est crée. Aussi, il défini les « properties sheets » de l'objet qui ajoute les méthodes « get » et « set » pour faire plus gérable l'objet. En plus, les différents types acceptés sont définis.

Dans SafeImage, il y a deux portal types, toutefois le plus important est Image Tile Transformed puisqu'il attache Zoomify dans ERP5, l'autre portal type Image Tile Group est en fait une petite modification du portal type Image , qui est déjà fournit pour ERP5 grâce au business template erp5_dms.

En analysant plus en détail le portal type Image Tile Transformed, sa classe définisse comme l'objet est crée. Ça veut dire qu'il définit comme l'image est coupé et comme les objets crées sont hébergés dans ERP5. En fait, cette classe fait le plus du travail. La classe s'appelle TileImageTransformed qui hérite de la classe de base Image et il a seulement une fonction qui est nettoyer le contenu de l'objet Tile Transformed, dans le cas que l'objet soit crée pour première fois, Zoomify est invoqué à travers de la fonctionne externe Image_getERP5Zoomify.

Alors même que Zoomify est adapté à Zope, plusieurs modifications ont été faites puisque Zope est seulement la base d'ERP5, et ceci ajoute plusieurs modifications qu'il faut avoir en compte pour le faire compatible avec Zoomify.

Approximativement, le « workflow » est le suivant :

- Créer le dossier qui contient toutes les « tiles » de l'image.
- Créer le conteneur pour le prochain groupe de « tile » dans le conteneur des données.
- Charger les données de l'image.
- Extraire les caractéristiques (metadata) de l'image.
- Garder chaque tile dans ERP5.
- Garder le fichier de transformations dans ERP5.
- Garder le fichier XML dans ERP5.

Selon cette « workflow », la structure créé est :

- ImageProperties.xml : contient la largeur, hauteur de l'image originale et la quantité totale de tiles.
- Transformations.txt : contient toutes les transformations qu'il faut appliquer à chaque tile.
- TileGroup : Dossier qui contient tous les tiles.
    - Tile (X-X-X) : Le « tile » lui-même. Le schéma de nommage est le même utilisé par Zoomify.

Donc, le portal type Image Tile Transformation supporte les types des données suivantes :

- Embedded File: Plain texte ou fichiers XML.
- Image
- Image Tile Group : Spécialement crée pour contenir les Images et les Embedded Files au même temps.

Ensuite, une vue est définie pour permettre aux utilisateur interagir avec ERP5. Dans ce cas, tous les deux portal types utilisent une vue très similaire à la vue du portal type Image avec l'exception que une listbox a été ajouté pour afficher les différent types d'éléments qu'il peut contenir. En plus de cette ressemblance, une champ générique a été ajouté pour assurer une bonne couplage avec les autres business templates.

Aussi, les scripts python ont un rôle important dans SafeImage puisque ils permettent les requêtes AJAX à côté-client. Sans eux, SafeImage n'aurait pas sens puisque le téléchargement dynamique des images ne serait pas possible. En fait, les scripts python sont le lien réelle entre le serveur et le client. Aussi, ils sont le plus facile et le plus vite moyenne d'échanger l'information. Dans SafeImage, Il y a trois scripts :

- ERP5Site_getTileImageTransformMetadaList : il fournit le fichier JSON qui contient la liste d'objets Image Tile Transformed hébergés dans ERP5.
- TileImage_getMetadaAsJSON : il retourne en format JSON, le fichier XML crée par Zoomify qui contient la largeur, la hauteur et la quantité de tiles.
- TileImageTransformed_getTransform : il retourne depuis le serveur, un fichier JSON avec les transformations qu'il faut appliquer à chaque tile de l'image.

Aussi, les scripts python doivent contourner la problématique de cross-domaine que les requêtes AJAX ont pour définition. En fait, le cross-domaine apparaît quand il s'agit d'échanger information asynchrone entre deux entités qui ont activés ses respectifs domaines de surveillance. Donc, il faut que les deux entités donnent les permis correspondants de sécurité. Autrement, la réponse AJAX serait vide et l'application ne marcherait pas.

Alors pour fixer cette problématique, la configuration de la sécurité à niveau de serveur (Zope) et client (code JavaScript) a été modifié.

3.1 Pixastic

Comme il est déjà annoncé, la librairie Pixastic a été utilisé pour la manipulation à niveau de pixel les images. Quelques algorithmes ont été adapté à OpenLayers. La manipulation des images est faite juste avant de l'affichage du « tile » au « canvas ». Cette processus a été fait pour montrer aux clients que SafeImage peut manipuler les images selon les besoins du client. Sur le prototype SafeImage présenté, il y a quelques algorithmes courants implémentés mais qu'il faudra les changer selon l'application. Par exemple : dans le projet TSXX qui travaille avec images satellitaires, l'élimination de bruit, le contraste et le gradient mathématique pour détecter changements pourraient être les algorithmes utilisés.

Vraiment, Pixastic est une libraire très complète et facile de gérer. Toutefois, quelques modifications ont été faites pour pouvoir utiliser Pixastic et OpenLayers ensemble.

III. Amélioration d'erp5testnode

1. Introduction

D'abord de tout, il s'agit d'expliquer pourquoi il y a besoin d'utiliser tests pendant le développement d'un logiciel.

En fait quand un logiciel devient très complexe et il est composé pour plusieurs modules ou milliards de lignes de code. Il s'agit d'avoir une procédure pour tester toutes les fonctionnalités d'une façon automatisé, puisque manuellement ce serait très lent et sûrement les résultats seraient pire.

Donc, les tests unitaires sont très importants pour assurer un correcte déploiement du logiciel. Ainsi comme, les test unitaires sont une garantie de qualité pour les clients puisque beaucoup de scénarios ont été déjà testés. Ainsi comme pour les développeurs, le travail est plus confortable puisque ils ne doivent pas vérifier les bugs de ses collègues de travail.

Dans SafeImage, il y a deux types différents de tests : les unitaires et les fonctionnelles. Les premières testent le code tandis que les autres simulent l'interaction d'une utilisateur avec le logiciel, chez Nexedi s'utilise le « framework » Selenium qui utilise le navigateur Firefox pour faire la simulation.

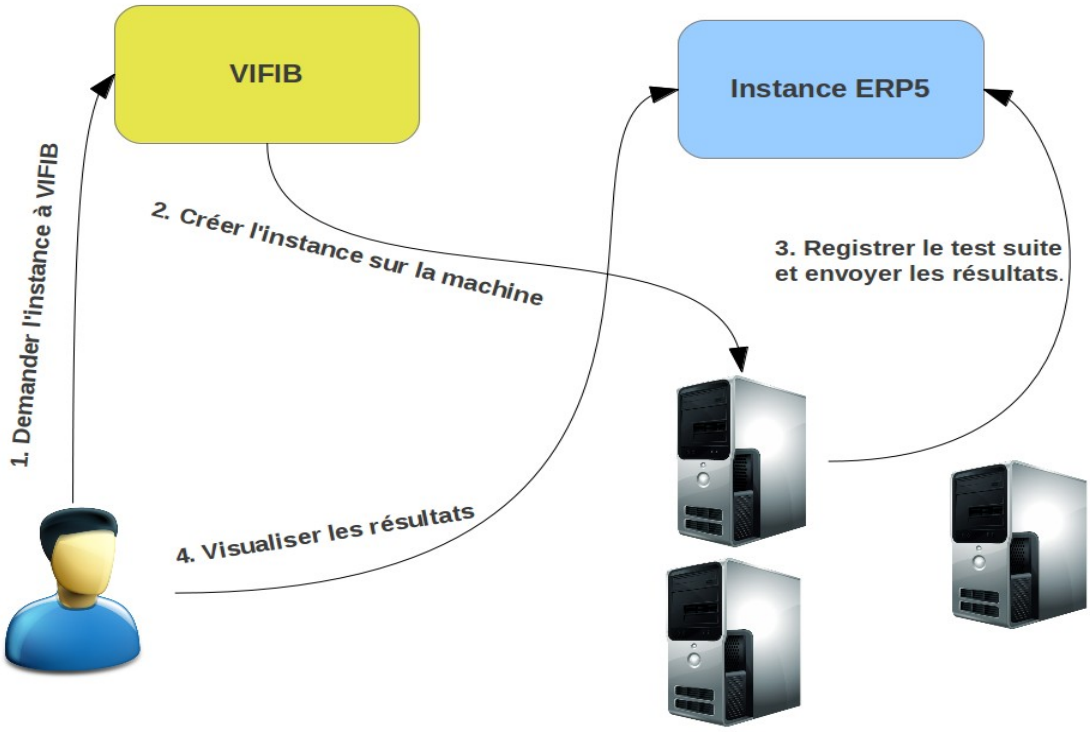
Chez Nexedi, le logiciel qui s'occupe de lancer les tests automatiquement s'appelle erp5testnode. En fait, erp5testnode est une « software release » qui permet lancer tests déjà codés et envoyer les résultats au destinataire désiré. En fait, erp5testnode est une version compressé d'ERP5, ça veut dire que la base de données, Zope et les autres logiciels dépendants sont installés aussi. Dans erp5testnode, chaque test est formé par au moins une test suite qui est un ensemble de paramètres utilisés qui décrivent l’environnent à tester.

Pour installer erp5testnode, il s'agit seulement d'aller à « www.vifib.com » et demander la création d'une instance erp5testnode. Finalement, la instance est créé automatiquement par slapos-node.

2. Optimisation d'erp5testnode

2.1 Ancienne structure

Pour pouvoir justifier les améliorations faites plus facilement, l'ancienne schéma est expliqué en détail. L'image suivante montre graphiquement les interactions qu'il y a pendant le processus.

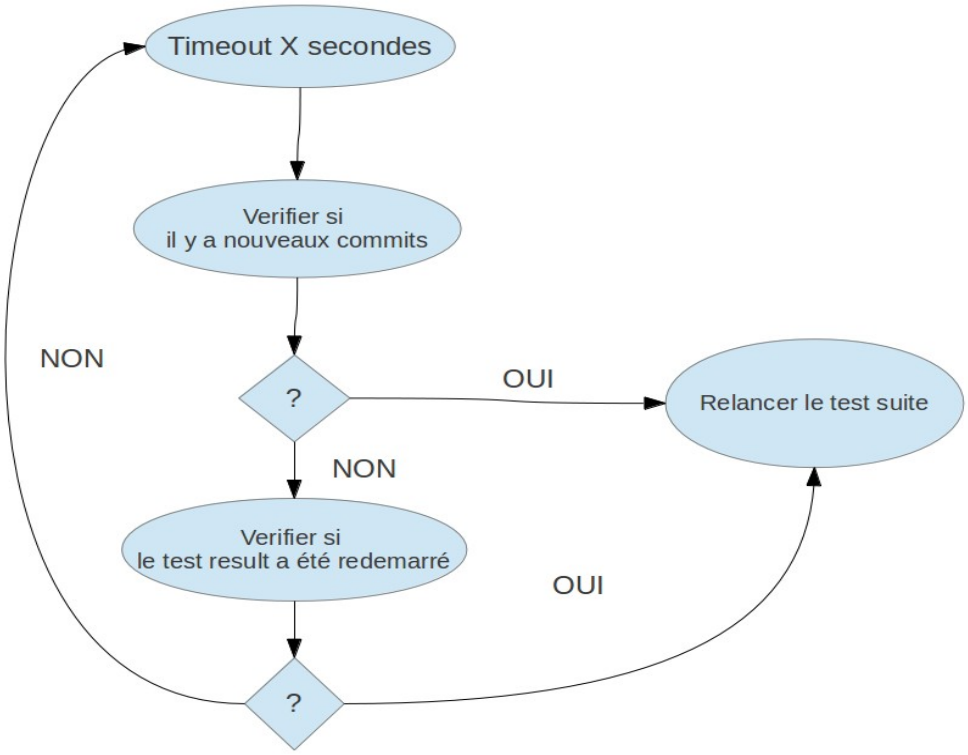


L'image ci-dessus montre une scénario courant à Nexedi où un développeur veut vérifier les modifications faites et il n'a pas aucun instance disponible.

D'abord de tout, il demande à VIFIB la création d'une instance erp5testnode. Ensuite, erp5testnode se connecte automatiquement à travers du protocole XMLRPC au serveur qui a installé l'instance ERP5. Il faut remarque que cette système n'est pas résilient puisque si quelque fois le serveur s'échoue, l'utilisateur ne peut pas voire les résultats, sauf qu'il avait accès SSH à la machine où le test tourne.

Le première échange d'information sert pour enregistrer le test suite dans ERP5 grâce à erp5_testresult qui gère toutes les résultats envoyés par les instances erp5testnode. Comme il est déjà expliqué, le test suite est un ensemble

paramètres qui définissent le tests à tourner. Par la suite, l'instance erp5testnode continue en envoyant les résultats des test déjà complétés jusqu'à il finisse. Alors, la boucle de l'image suivante est répétée indéfiniment.



Selon l'image ci-dessus, la structure utilisé n'est pas efficient puisque quand le test suite finit, l'instance est inactivé jusqu'à l'utilisateur a besoin d'un nouveau test. Donc, les ressources dédiée ne sont pas utilisé à temps complet.

Aussi, le code lui-même a quelques bugs qui font que l'administrateur des tests nodes de l’entreprise dépense trop de temps pour activer et désactiver les instances. Par exemple : pendant le stage, j’étais l'administrateur des test nodes et il y avait beaucoup de fois que les développeurs ont moi demandé de démarrer les instances puisqu'ils ont était bloqué.

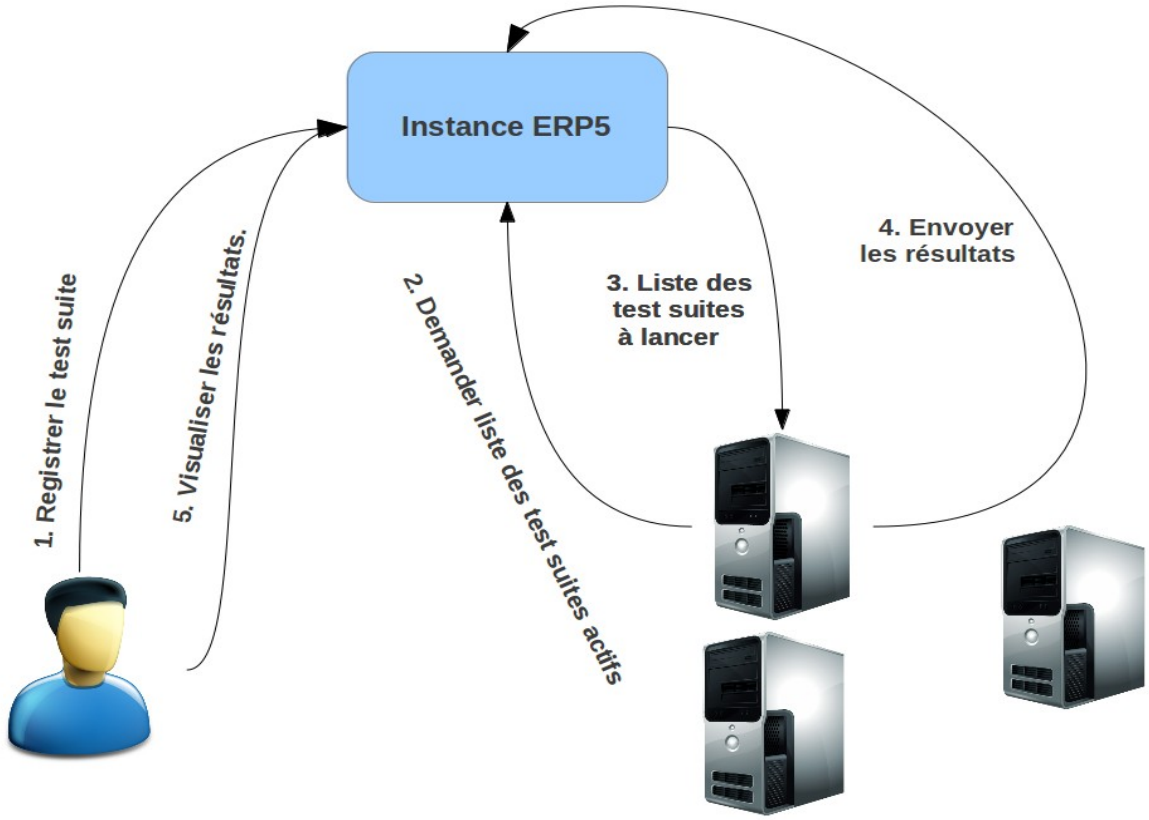
En conclusion, l'ancienne structure utilisé à Nexedi était bonne mais il a fallu la faire plus automatisé pour pouvoir être vraiment utile pour l’entreprise.

2.2 Nouveau structure

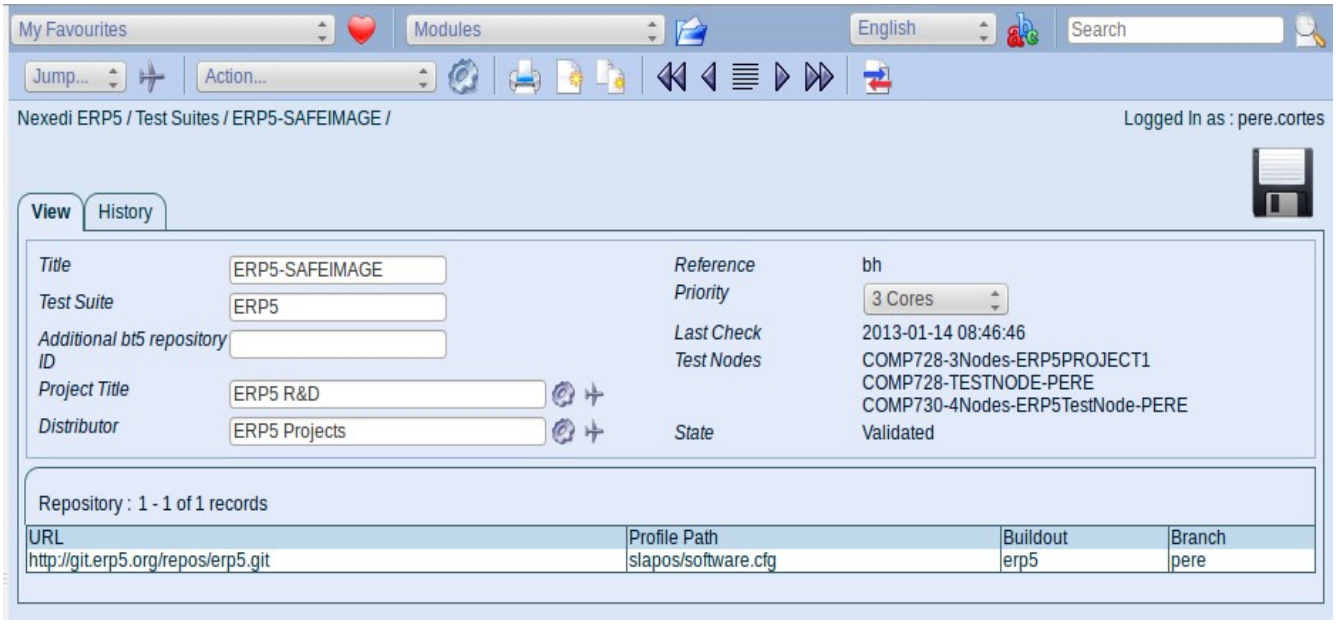
Comme il est déjà expliqué, le testage à Nexedi n'était pas assez bonne pour être utile pour les développeurs. Donc quelques modifications ont été fait pour assurer une bonne fonctionnement. Les principales modifications faites sont :

- Création d'un nouveau business template dans ERP5. Cette business template gère toutes les test suites disponibles dans ERP5 et il les alloue automatiquement aux instances erp5_testnode
- Modification de le « egg » erp5_util pour optimiser la performance et faire plus facile l'utilisation pour les développeurs.
- Élimination de le couplage entre les test suites et les instances. Ça veut dire que chaque instance peut tourner différents test suites.
- Création d'un algorithme de décision pour associer à chaque instance les test suite à tourner.
- Création de test unitaires pour vérifier le fonctionnement correct de erp5_testnode lui-même. Avant, il n'y avait pas.

L'image ci-dessous montre graphiquement les interactions :

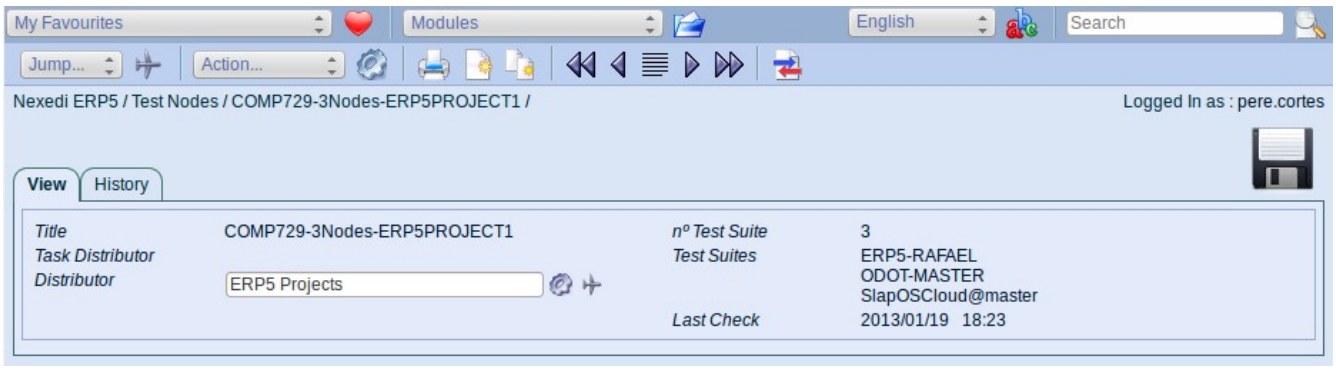


L'image ci-dessus montre comme désormais, l'utilisateur ne doit pas créer sa propre instance sinon il doit aller à l'instance ERP5 et enregistrer sa test suite sur le nouveau module test node. Alors, le test suite est associé à une instance erp5testnode qui a été déjà enregistré sur ERP5. Il faut remarque qu'un test suite peut tourner sur différentes instances au même temps. Ça fait que les tests suites finissent tôt et le couplage entre les test suites et les instances erp5_testnode soit moindre. Puisque si un instance erp5testnode échoue le test suite tournera sur une autre instance.



L'image ci-dessus montre le test suite ERP5-SAFEIMAGE qui teste le projet SafeImage. Le nouveau module affiche les instances où le test suite est en train de tourner. Il faut remarquer que la quantité d'instances dépend du champ « Priority ». Donc grâce à l'algorithme ajouté, chaque test suite a une priorité qui détermine la quantité d'instances différents où le test suite tournera. Ça donne certain liberté aux développeur pour choisir l'importance du test suite et à niveau d'entreprise, les projets sont priorisé selon son importance.

Néanmoins, avant d'enregistrer le test suite, il s'agit d'enregistrer l'instance erp5testnode (le test node) sur ERP5. C'est fait en utilisant le nouveau module « Test Node ». Toutefois, c'est fait pour les administrateurs des test nodes.



L'image ci-dessus, montre un test node quelconque enregistré dans ERP5. Le champ « Last Check » est pour contourner le possible échoué d'un test node ou problèmes avec le réseau. En fait, « Last Check » est mis à jour chaque fois que l'instance test node se connecte au serveur ERP5. Chaque 10 minutes, ERP5 lance automatiquement une alarme qui verifie que la différence entre le temps de vérification est le Last Check est plus petite que dix heures. Sinon, le test node est mis sur l'état « Invalidated ». Ça veut dire que désormais aucune test suite sera assigné au test node.

Donc, l'utilisateur doit seulement aller au module « test result » d'ERP5 et vérifier si les résultats sont correctes. L'image suivante montre les résultats obtenues pendant le test de SafeImage.

Nexedi ERP5 / Test Results / ERP5-SAFEIMAGE /

Logged In as : pere.cortes

View | Nodes | History

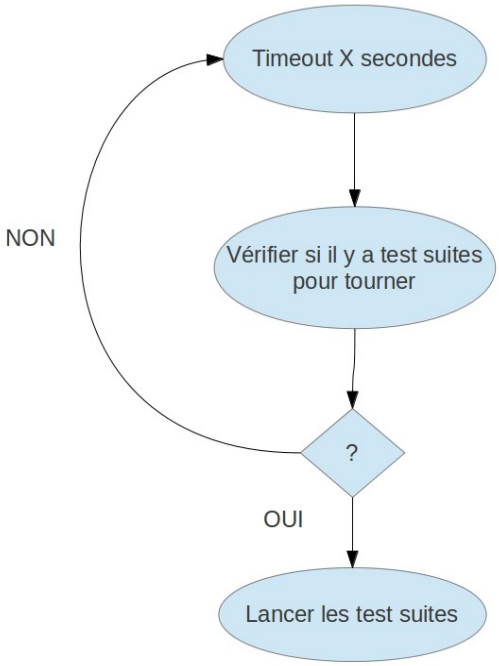
Title	ERP5-SAFEIMAGE	Project	Resilience Project
Reference	erp5=41634-ad77352184b5ddbfa7a0ae366ce908cd80e98b06	All Tests	3493
Test Report ID	20130111-DC0D29C	Failures	0
Build Slave Name		Errors	0
Revision		Skips	195
Launch Date	2013/01/11 14:43	Test Result	PASS
Completion Date	2013/01/11 20:22	Test Status	Completed

Comment

Test Results : 1 - 0 of 260 records

Test Case	Start Date	Duration	All Tests	Errors	Failures	Skips	Result	Status
testAccounting	2013/01/11 16:53:2.478080 GMT	1528.202	85	0	0	0	PASSED	Completed
testAccountingReports	2013/01/11 17:46:50.752023 GMT	452.485	58	0	0	1	PASSED	Completed
testAccountingRules	2013/01/11 19:45:49.335157 GMT	258.331	22	0	0	22	PASSED	Completed
testAcknowledgementTool	2013/01/11 20:04:18.417085 GMT	66.767	1	0	0	0	PASSED	Completed
testAdvancedInvoicing	2013/01/11 17:46:44.222653 GMT	565.27	24	0	0	4	PASSED	Completed
testAlarm	2013/01/11 20:03:57.537936 GMT	147.192	24	0	0	0	PASSED	Completed
testAmount	2013/01/11 19:45:9.294962 GMT	154.264	57	0	0	0	PASSED	Completed
testApparelModel	2013/01/11 19:49:59.377560 GMT	105.346	2	0	0	0	PASSED	Completed
testApparelTransformation	2013/01/11 19:40:0.573945 GMT	377.68	1	0	0	0	PASSED	Completed
testArchive	2013/01/11 19:35:32.320965 GMT	187.364	1	0	0	0	PASSED	Completed
testAudioField	2013/01/11 20:18:21.114783 GMT	102.421	1	0	0	0	PASSED	Completed
testAuthenticationPolicy	2013/01/11 20:06:50.285352 GMT	94.885	6	0	0	0	PASSED	Completed
testAutoLogout	2013/01/11 20:04:15.982230 GMT	61.144	1	0	0	0	PASSED	Completed
testBPMCore	2013/01/11 19:46:29.100851 GMT	379.63	12	0	0	1	PASSED	Completed
testBackportUnittest	2013/01/11 20:22:12.794795 GMT	0.001	7	0	0	0	PASSED	Completed
testBase	2013/01/11 20:14:16.001123 GMT	176.051	36	0	0	3	PASSED	Completed
testBudget	2013/01/11 19:24:5.826891 GMT	172.311	21	0	0	0	PASSED	Completed
testBug	2013/01/11 19:55:59.080936 GMT	123.315	9	0	0	0	PASSED	Completed
testBusinessTemplate	2013/01/11 16:42:10.915353 GMT	5787.696	106	0	0	5	PASSED	Completed
testBusinessTemplateInstallation	2013/01/11 19:43:42.123132 GMT	126.458	1	0	0	0	PASSED	Completed
testCMFActivity	2013/01/11 20:01:5.851080 GMT	77.218	113	0	0	0	PASSED	Completed
testCMFCategory	2013/01/11 19:52:59.144121 GMT	215.923	46	0	0	0	PASSED	Completed
testCRM	2013/01/11 18:19:11.273803 GMT	312.145	47	0	0	1	PASSED	Completed
testCSSPacker	2013/01/11 20:22:12.578360 GMT	0.004	4	0	0	0	PASSED	Completed
testCache	2013/01/11 20:00:20.482251 GMT	96.647	3	0	0	0	PASSED	Completed
testCacheTool	2013/01/11 19:58:44.020220 GMT	90.841	7	0	0	0	PASSED	Completed
testCachedSkinsTool	2013/01/11 20:16:33.281465 GMT	98.404	5	0	0	0	PASSED	Completed
testCalendar	2013/01/11 19:59:16.805204 GMT	102.875	17	0	0	0	PASSED	Completed

Quand toutes les test suites assignées sont déjà fait, l'instance erp5testnode entre dans la boucle suivante:



Il s'agit de remarquer que la boucle est maintenant plus simple car le test node doit seulement vérifier si il y a plus de tests suites à tourner. La simplicité réside sur le fait qu'il y aura plus qu'un test suite à tourner. Donc, la probabilité que le test node soit inactive est plus petite qu'avant.

Pour conclure, les modifications ajoutées ont impact sur différents aspects de la performance du test node. D'abord de tout à niveau d'utilisateur, l'interaction est plus facile puisque il doit seulement enregistré le test suite sur ERP5 quand avant, il a fallu aller à VIFIB et créer une instance spécifiquement pour son test suite. Aussi, les ressources sont plus optimisés car les ordinateurs tournent plus de test suites ensemble car les ordinateurs ont plus de test à tourner, a moins les mêmes qu'avant. Dans le bureau, mon chef et moi ont vérifié ça puisqu'il a fait plus de chaud qu'avant et en vérifiant l'utilisation de la « CPU » des ordinateurs. Par rapport à l'électricité, les ordinateurs dépensent moins d'énergie parce que l'optimisation est plus grand. Donc dans les futures projets, Nexedi pourra assurer à ses client moins de consommation d'électricité. Ainsi comme, les ordinateurs auront plus de mémoire libre pour gérer autres choses.

À niveau de la structure « cloud », maintenant il y a un « cloud » plus transparent puisque l'utilisateur ne saura pas où les tests tournent. Toutefois, il y a encore un structure « cloud » centralisé puisque si le serveur central ERP5 échoue, les résultats ne pourront pas être afficher. Donc, c'est un possible amélioration pour

le future prochain. Néanmoins, Nexedi n'est pas préoccupé pour cette aspect puisque les tests sont utilisé pour les développeurs et si ils ont quelque problème de connectivité, ils peuvent demander l’accès pour SSH à l’administrateur des tests nodes.
En fait mathématiquement, la performance et l'optimisation est mieux chaque fois qu'un test suite est ajouté parce que les ordinateurs auront plus de tests suites à tourner et l'activité sera plus grand par rapport à l'ancienne structure qui serait inactif plus de temps.

Conclusion

Le stage a été très enrichissant et très instructifs sur de nombreux aspects. Tout d'abord au niveau de la programmation car il s'agissait de ma première expérience de programmation dans un univers professionnel et en plus quand le niveau de programmation à Nexedi est vraiment très haut. Aussi, la découverte des langages web, et en particulier du JavaScript, a été très riche. Ainsi comme, connaître le langage Python m'a permis de voire la puissance des langages à côté-serveur et sa importance dans le développement d'un logiciel. Utiliser un système ERP5 m'a permis de mieux comprendre le fonctionnement d'un grand entreprise fonctionne.

Aussi, J'étais très surpris et séduit par l'entreprise Nexedi. Son Business model, en particulier, basé sur l'implémentation d'une application open-source et pour sa lutte pour les droits des utilisateurs. Enfin, les nombreux rapports que J'ai pu avoir avec l'ensemble du personnel de l'entreprise ont aussi participé à mon développement personnel. Ainsi comme, J'ai fait beaucoup d'amis et collègues dans l'entreprise qui a permis que J'allais plus content à travailler.

Concernant le travail fourni, je suis particulièrement satisfait du travail fait pendant mon stage. Concernant SafeImage, les objectifs ont été clairement accomplis. Toutefois, Je suis commencé depuis zéro et la problématique avec le cross domaine qui m'a fait passer beaucoup de temps pour le fixer. Mais grâce à l'aide de mon chef Seb, le projet SafeImage s'est déroulé très bien. Concernant au test node, J'ai souffert un peu manque d'expérience sur un logiciel déjà implémenté. Ainsi comme, la pression de vérifier toutes les erreurs avant de mettre à jour le logiciel a fait que parfois J'étais très nerveuse et Je faisais erreurs méchants. Néanmoins, mon chef m'a toujours transmis tranquillité et sérénité pour affronter mes problèmes. C'était très utile pour mon avenir puisque désormais, Je serais plus organisé et plus patient pour trouver la solution du problème.

Globalement, le stage m'a permis de connaître beaucoup de nouveaux choses et surtout d'approfondir sur thèmes déjà étudiés à l'école mais à un niveau plus haut. En fait, mon stage a été un mélange des domaines. Puisque, J'ai fait programmation, algorithmes d'images et réseau. Trois domaines très important pour le « cloud » qui a toujours été mon principal intérêt. En plus, pouvoir voire comme le travail fait sur le test node complémente et améliore mon projet SafeImage a été très enrichissant et satisfaisant.

Aussi, Je souhaite que Nexedi puisse implémenter et vendre mon projet aux clients tôt car Je crois que démontrer la possibilité de télécharger images larges depuis dispositifs qui ont une faible puissance. C'est un avance très grand dans le domaine des ces dispositifs. Je suis vraiment sure que cette projet sera très important pour Nexedi pendant les prochaines années.

Bibliographie

- JavaScript: The Good Parts, par Douglas Crockford
- Dive Into Python par Mark Pilgrim
- Dive Into HTML5 par Mark Pilgrim
- Code Convention for the JavaScript Programming Language, par Douglas Crockford : http://javascript.crockford.com/code.html
- Classical Inheritance in JavaScript, par Douglas Crockford: http://javascript.crockford.com/inheritance.html
- Prototypal Inheritance in JavaScript, par Douglas Crockford: http://javascript.crockford.com/prototypal.html
- Site officiel de Pixastic : http://www.pixastic.com
- Site officiel de IcanHazJs : http://www.icanhazjs.com
- Conseils de mozilla fondation sur le Cross-domaine : https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS
- Site officiel de Zoomify : http://www.zoomify.com
- Site officiel de jQuery : http://www.jquery.com

Annexe

TileImageTransformed.py

```
from Products.ERP5.Document.Image import Image
from zLOG import LOG,INFO,ERROR,WARNING

class TileImageTransformed(Image):
  """
  Tile Images split images in many small parts and then store informations as sub objects
  """
  def _setFile(self, *args, **kw):
    """Set the file content and reset image information."""
    if "TileGroup0" in self.objectIds():
        self.manage_delObjects("TileGroup0")
    if "ImageProperties.xml" in self.objectIds():
        self.manage_delObjects("ImageProperties.xml")
    self._update_image_info()
    processor = self.Image_getERP5ZoomifyProcessor(self,True)
    processor.ZoomifyProcess(self.getId(),*args)
```

Classe ERP5ZoomifyZopeProcessor pour adapter Zoomify à ERP5

```
class ERP5ZoomifyZopeProcessor(ZoomifyZopeProcessor):

  def __init__(self, document,transformed=None):
     self.document = document
     self.transformed = transformed
     self.count = 0

  def createTileContainer(self, tileContainerName=None):
    """ create each TileGroup """

    self.document.newContent(portal_type="Image Tile Group",
                title=tileContainerName, id=tileContainerName,
                filename=tileContainerName)
    return

  def createDefaultViewer(self):
```

```
  """ add the default Zoomify viewer to the Zoomify metadata """
  pass
  return

def createDataContainer(self, imageName="None"):
  """Creates nothing coz we are already in the container"""
  pass
  return

def _updateTransformedFile(self,tile_group_id,tile_title):
  """create and save the transform file"""
  num = random.choice([0,1])
  while num >= 0:
    algorithm = random.choice(['sepia','brightness','noise','lighten',
                  'posterize','edge','none'])
    if algorithm == 'lighten':
      param1 = random.choice([-0.6,-0.5,-0.4,-0.3,-0.2,-0.1,0.1,0.2,
                  0.3,0.4,0.5,0.6])
      param2 = 0
    elif algorithm == 'posterize':
      param1 = random.choice([4,5,6,7,8,9,10,11,12,13,14,15,16,17,
                  18,19,20,21])
      param2 = 0
    elif algorithm == 'brightness':
      param1 = random.choice([-80,-60,-40,-20,20,40,60,80])
      param2 = random.choice([-0.3,-0.2,-0.1,0,0.1,0.5,0.9])
    else:
      param1 = 0
      param2 = 0
    my_text = '%s %s %s %s %s %s \n' %(tile_group_id, tile_title,
                  algorithm, param1, param2, num)
    self.my_file.write(my_text)
    num = num - 1


def saveTile(self, image, scaleNumber, column,row):
  """save the cropped region"""
  tileFileName = self.getTileFileName(scaleNumber, column, row)
  tileContainerName = self.getAssignedTileContainerName(
                  tileFileName=tileFileName)
```

```
    namesplit = tileFileName.split('.')
    w,h = image.size
    if w != 0 and h !=0:
      tile_group_id = self.getAssignedTileContainerName()
      tile_group=self.document[tile_group_id]
      tileImageData= StringIO()
      image.save(tileImageData, 'JPEG', quality=self.qualitySetting)
      tileImageData.seek(0)
      if tile_group is None:
        raise AttributeError('unable to fine tile group %r' % tile_group_id)
      w = tile_group.newContent(portal_type='Image', title=namesplit[0],
              id=namesplit[0], file=tileImageData, filename=namesplit[0])
      if self.transformed:
        self._updateTransformedFile(tile_group_id, namesplit[0])
    return

  def saveXMLOutput(self):
    """save the xml file"""
    my_string = StringIO()
    my_string.write(self.getXMLOutput())
    my_string.seek(0)
    self.document.newContent(portal_type='Embedded File',
                        id='ImageProperties.xml', file=my_string,
                        filename='ImageProperties.xml')


      return

  def saveTransformedFile(self):
    """add in Zope the transform file """
    if self.transformed:
      self.my_file.seek(0)
      self.document.newContent(portal_type='Embedded File',
                        id='TransformFiletxt', file=self.my_file,
                        filename='TransformFiletxt')
    return


def getERP5ZoomifyProcessor(document,transformed=False):
  return ERP5ZoomifyZopeProcessor(document,transformed)
```

testSafeImage pour tester automatiquement SafeImage

```
import Image
from Products.ERP5Type.tests.ERP5TypeTestCase import ERP5TypeTestCase
import transaction
from zLOG import LOG,INFO,ERROR
import json
from cStringIO import StringIO
import os

class FileUpload(file):
  """Act as an uploaded file.
  """
  __allow_access_to_unprotected_subobjects__ = 1
  def __init__(self, path, name):
    self.filename = name
    file.__init__(self, path)
    self.headers = {}

def makeFilePath(name):
  #return os.path.join(os.path.dirname(__file__), 'tmp', name)
  return name

def makeFileUpload(name, as_name=None):
  if as_name is None:
    as_name = name
  path = makeFilePath(name)
  return FileUpload(path, as_name)

class TestSafeImage(ERP5TypeTestCase):

  def getBusinessTemplateList(self):
    return ('erp5_base',
            'erp5_web',
            'erp5_ingestion_mysql_innodb_catalog',
            'erp5_ingestion',
            'erp5_dms',
            'erp5_safeimage'
           )
```

```
 def afterSetUp(self):
   portal = self.getPortalObject()
   self.image_module = self.portal.getDefaultModule(portal_type = 'Image Module')
   self.assertTrue(self.image_module is not None)
   if getattr(self.image_module,'testImage',None) is not None:
     self.image_module.manage_delObjects(ids=['testImage'])
   if getattr(self.image_module,'testTile',None) is not None:
     self.image_module.manage_delObjects(ids=['testTile'])
   if getattr(self.image_module,'testTileTransformed',None) is not None:
     self.image_module.manage_delObjects(ids=['testTileTransformed'])
   transaction.commit()
   self.tic()

 def _createImage(self):
   portal = self.getPortalObject()
   image = portal.restrictedTraverse('portal_skins/erp5_safeimage/img/image_unit_test.jpg')
   path_image = "image_unit_test.jpg"
   fd = os.open(path_image, os.O_CREAT| os.O_RDWR)
   os.write(fd,str(image.data))
   os.close(fd)
   _image = makeFileUpload(path_image)
   image = self.image_module.newContent(portal_type='Image',title='testImage',
                           id='testImage',file=_image,filename='testImage')
   return image

 def _createTileImage(self):
   portal = self.getPortalObject()
   image = portal.restrictedTraverse('portal_skins/erp5_safeimage/img/image_unit_test.jpg')
   path_image = "image_unit_test.jpg"
   fd = os.open(path_image, os.O_CREAT| os.O_RDWR)
   os.write(fd,str(image.data))
   os.close(fd)
   tile_image = makeFileUpload(path_image)
   tile = self.image_module.newContent(portal_type='Image Tile',title='testTile',
                          id='testTile',file=tile_image,filename='testTile')
   return tile

def _createTileImageTransformed(self):
   portal = self.getPortalObject()
```

```
    image = portal.restrictedTraverse('portal_skins/erp5_safeimage/img/image_unit_test.jpg')
    path_image = "image_unit_test.jpg"
    fd = os.open(path_image, os.O_CREAT| os.O_RDWR)
    os.write(fd,str(image.data))
    os.close(fd)
    tile_image_transformed = makeFileUpload(path_image)
    tile_transformed = self.image_module.newContent(portal_type='Image Tile Transformed',
                          title='testTileTransformed',id='testTileTransformed',
                          file=tile_image_transformed,filename='testTileTransformed')
    return tile_transformed

  def test_01_CreateImage(self):
    image = self._createImage()
    self.assertTrue(image.hasData())
    transaction.commit()
    self.tic()
    self.assertNotEqual(image,None)

def test_02_CreateTileImage(self):
    """
    We are going to check that tile image has following structure
    1/
    1/Image Tile Group
    1/Image Tile Group/0-0-0
    1/Image Tile Group/1-0-0
    1/ImageProperties.xml
    """
    tile = self._createTileImage()
    transaction.commit()
    self.tic()
    self.assertNotEqual(tile,None)
    image_property = getattr(tile, "ImageProperties.xml", None)
    self.assertEquals(image_property.getData(),
 """<IMAGE_PROPERTIES WIDTH="660" HEIGHT="495" NUMTILES="9" NUMIMAGES="1" VERSION="1.8"
TILESIZE="256" />""")
    self.assertNotEqual(image_property, None)
    self.assertEquals("Embedded File", image_property.getPortalType())
    image_group = getattr(tile, "TileGroup0", None)
    self.assertNotEquals(image_group, None)
    self.assertEquals("Image Tile Group",image_group.getPortalType())
```

```
    splitted_image_list = image_group.objectValues(portal_type="Image")
    self.assertEquals(set(['0-0-0','1-0-0','1-1-0','2-0-0','2-0-1','2-1-0','2-1-1','2-2-0','2-2-1']),
                      set([x.getId() for x in splitted_image_list]))
    for x in splitted_image_list:
      self.assertTrue(x.hasData())
    self.assertEquals(123,image_group['0-0-0'].getHeight())
    self.assertEquals(165,image_group['0-0-0'].getWidth())

def test_03_CreateTileImageTransformed(self):
    """
    We are going to check that tile image has following structure
    1/
    1/Image Tile Group
    1/Image Tile Group/0-0-0
    1/Image Tile Group/1-0-0
    1/ImageProperties.xml
    1/TransformFiletxt
    """
    tile_transformed = self._createTileImageTransformed()
    transaction.commit()
    self.tic()
    self.assertNotEqual(tile_transformed,None)
    image_property = getattr(tile_transformed, "ImageProperties.xml", None)
    self.assertEquals(image_property.getData(),
 """<IMAGE_PROPERTIES WIDTH="660" HEIGHT="495" NUMTILES="52" NUMIMAGES="1" VERSION="1.8"
TILESIZE="256" />""")
    self.assertNotEqual(image_property, None)
    self.assertEquals("Embedded File", image_property.getPortalType())
    image_transform = getattr(tile_transformed, "TransformFiletxt", None)
    self.assertTrue(image_transform.getData().split()[1],'2-0-0')
    self.assertNotEqual(image_transform, None)
    self.assertEquals("Embedded File", image_transform.getPortalType())
    image_group = getattr(tile_transformed, "TileGroup0", None)
    self.assertNotEquals(image_group, None)
    self.assertEquals("Image Tile Group",image_group.getPortalType())
    splitted_image_list = image_group.objectValues(portal_type="Image")
    self.assertEquals(set(['0-0-0','1-0-0','1-1-0','2-0-0','2-0-1','2-1-0','2-1-1','2-2-0','2-2-1']),
                      set([x.getId() for x in splitted_image_list]))
    for x in splitted_image_list:
      self.assertTrue(x.hasData())
```

```
    self.assertEquals(123,image_group['0-0-0'].getHeight())
    self.assertEquals(165,image_group['0-0-0'].getWidth())
    if getattr(self.image_module,'testTileTransformed',None) is not None:
     self.image_module.manage_delObjects(ids=['testTileTransformed'])
    transaction.commit()
    self.tic()
```

form.js pour gèrer les requêtes AJAX entre le serveur et le client

```
/**
 * NEXEDI
 */
(function($) {

  $.getJSON(
    'http://'+window.location.host+'/erp5/ERP5Site_getTileImageTransformMetadataList',
     function(data){
         for (var i = 0; i < data["image_list"].length; i ++ ) {

                  var aux1= "<li><a href=#image/";
                  var aux2= "><i class=icon-star></i>";
                  var aux3= "</a></li>";
           $('.nav-header').append(aux1+data["image_list"][i]["id"]+aux2+data["image_list"][i]["title"]
+aux3)

         };
      });

 var routes = {
  "/image/:id" : "displayData",
  "image/:id" : "displayData",
 }

 var router = function(e, d){
  var $this = $(this);
  $.each(routes, function(pattern, callback){
    pattern = pattern.replace(/:\w+/g, '([^\/]+)');
    var regex = new RegExp('^' + pattern + '$');
    var result = regex.exec(d);
    if (result) {
```

```
      result.shift();
      methods[callback].apply($this, result);
    }
  });
}

var methods = {
  init: function() {
    // Initialize in this context
    var $this = $(this);
    // Bind to urlChange event
    return this.each(function(){
      $.subscribe("urlChange", function(e, d){
        router.call($this, e, d);
      });
    });
  },

  displayData: function(id){
    var zoomify_url, zoomify_width, zoomify_height = null;
    zoomify_url = "http://"+window.location.host+"/erp5/image_module/" + id + "/";
    //XXX look at the xml definition inside image folder
    var zoomify_data = $.getJSON(
                                    "http://"+window.location.host+"/erp5/image_module/" + id +
"/TileImage_getMetadataAsJSON",
                                    function(data){
                                            width=data["sizes"][0]["width"];
                                            height=data["sizes"][0]["height"];
                                     transforms(width,height);
                                    }

                                );

          $(this).form('render', 'image', {'image_id': id});


var transforms = function(width,height){
                $.getJSON(

'http://'+window.location.host+'/erp5/image_module/'+id+'/TileImageTransformed_getTransform',
```

```
                    function(data){
                      pass(width,height,data);
                     }
                   );
   }

          var pass = function(zoomify_width,zoomify_height,data){

                                        $(function() {
                                SafeImage.loadOpenLayerZoomedImage(zoomify_width,zoomify_height,
zoomify_url,data);
             if (document.location.search != ""){
               SafeImage.map.zoomTo(Number(document.location.search.split("")[6]));
              }
                                         });
          };

   },

   render: function(template, data){
           $(this).html(ich[template](data, true));
    }

      };

  $.fn.form = function(method){
   if ( methods[method] ) {
     return methods[method].apply( this, Array.prototype.slice.call( arguments, 1 ));
   } else if ( typeof method === 'object' || ! method ) {
     return methods.init.apply( this, arguments );
   } else {
     $.error( 'Method ' +  method + ' does not exist on jQuery.form' );
   }
  };
})(jQuery);

$("#main").form();
```

Test Unitaire pour erp5testnode

```
from unittest import TestCase
from erp5.util.testnode.testnode import TestNode
from erp5.util.testnode.testnode import SlapOSInstance
from erp5.util.testnode.ProcessManager import ProcessManager, SubprocessError

from erp5.util.testnode.SlapOSControler import SlapOSControler
from erp5.util.taskdistribution import TaskDistributor
from erp5.util.taskdistribution import TaskDistributionTool
from erp5.util.taskdistribution import TestResultProxy
import os
import shutil
import subprocess
import tempfile
import json
import time

class ERP5TestNode(TestCase):

  def setUp(self):
    self._temp_dir = tempfile.mkdtemp()
    self.working_directory = os.path.join(self._temp_dir, 'testnode')
    self.slapos_directory = os.path.join(self._temp_dir, 'slapos')
    self.test_suite_directory = os.path.join(self._temp_dir,'test_suite')
    self.environment = os.path.join(self._temp_dir,'environment')
    self.log_directory = os.path.join(self._temp_dir,'var/log')
    self.log_file = os.path.join(self.log_directory,'test.log')
    self.remote_repository0 = os.path.join(self._temp_dir, 'rep0')
    self.remote_repository1 = os.path.join(self._temp_dir, 'rep1')
    self.remote_repository2 = os.path.join(self._temp_dir, 'rep2')
    os.mkdir(self.working_directory)
    os.mkdir(self.slapos_directory)
    os.mkdir(self.test_suite_directory)
    os.mkdir(self.environment)
    os.makedirs(self.log_directory)
    os.close(os.open(self.log_file,os.O_CREAT))
    os.mkdir(self.remote_repository0)
    os.mkdir(self.remote_repository1)
```

```
    os.mkdir(self.remote_repository2)
    def log(*args,**kw):
      for arg in args:
        print "TESTNODE LOG : %r" % (arg,)
    self.log = log

  def tearDown(self):
    shutil.rmtree(self._temp_dir, True)

  def getTestNode(self):
    # XXX how to get property the git path ?
    config = {}
    config["git_binary"] =
"/srv/slapgrid/slappart80/srv/runner/software/ba1e09f3364989dc92da955b64e72f8d/parts/git/bin/git"
    config["slapos_directory"] = config["working_directory"] = self.working_directory
    config["node_quantity"] = 3
    config["test_suite_directory"] = self.test_suite_directory
    config["environment"] = self.environment
    config["log_directory"] = self.log_directory
    config["log_file"] = self.log_file
    config["test_suite_master_url"] = None
    config["test_node_title"] = "Foo-Test-Node"
    return TestNode(self.log, config)

  def getTestSuiteData(self, add_third_repository=False, reference="foo"):
    data = [{
      "test_suite": "Foo",
      "project_title": reference,
      "test_suite_title": "Foo-Test",
      "test_suite_reference": reference,
      "vcs_repository_list": [
          {'url': self.remote_repository0,
           'profile_path': 'software.cfg',
           'branch': 'master'},
          {'url': self.remote_repository1,
           'buildout_section_id': 'rep1',
           'branch': 'master'}]}]
    if add_third_repository:
      # add a third repository
      # insert in position zero since we already had bug when the profile_path
```

```
    # was defined in non-zero position when generating the profile
    data[0]['vcs_repository_list'].insert(0,
        {'url': self.remote_repository2,
         'buildout_section_id': 'rep2',
         'branch': 'foo'})
  return data

def updateNodeTestSuiteData(self, node_test_suite,
                            add_third_repository=False):
  node_test_suite.edit(working_directory=self.working_directory,
    **self.getTestSuiteData(add_third_repository=add_third_repository)[0])

def getCaller(self, **kw):
  class Caller(object):

    def __init__(self, **kw):
      self.__dict__.update(**kw)

    def __call__(self, command):
      return subprocess.check_output(command, **self.__dict__)
  return Caller(**kw)

def generateTestRepositoryList(self, add_third_repository=False):
  commit_dict = {}
  repository_list = [self.remote_repository0, self.remote_repository1]
  if add_third_repository:
    repository_list.append(self.remote_repository2)
  for i, repository_path in enumerate(repository_list):
    call = self.getCaller(cwd=repository_path)
    call("git init".split())
    call("touch first_file".split())
    call("git add first_file".split())
    call("git commit -v -m first_commit".split() + ['--author="a b <a@b.c>"'])
    my_file = open(os.path.join(repository_path, 'first_file'), 'w')
    my_file.write("initial_content%i" % i)
    my_file.close()
    call("git commit -av -m next_commit".split() + ['--author="a b <a@b.c>"'])
    output = call(['git', 'log', '--format=%H %s'])
    output_line_list = output.split("\n")
    self.assertEquals(3, len(output_line_list))
```

```
    # remove additional return line
    output_line_list = output_line_list[0:2]
    expected_commit_subject_list = ["next_commit", "first_commit"]
    commit_subject_list = [x.split()[1] for x in output_line_list]
    self.assertEquals(expected_commit_subject_list, commit_subject_list)
    commit_dict['rep%i' % i] = [x.split() for x in output_line_list]
    if repository_path == self.remote_repository2:
      output = call('git checkout master -b foo'.split())
  # commit_dict looks like
  # {'rep1': [['6669613db7239c0b7f6e1fdb82af6f583dcb3a94', 'next_commit'],
  #           ['4f1d14de1b04b4f878a442ee859791fa337bcf85', 'first_commit']],
  #  'rep0': [['fb2a61882148d705fd10ecd87278b458a59920a9', 'next_commit'],
  #           ['4f1d14de1b04b4f878a442ee859791fa337bcf85', 'first_commit']]}
  return commit_dict

def test_01_getDelNodeTestSuite(self):
  """
  We should be able to get/delete NodeTestSuite objects inside test_node
  """
  test_node = self.getTestNode()
  node_test_suite = test_node.getNodeTestSuite('foo')
  self.assertEquals(0, node_test_suite.retry_software_count)
  node_test_suite.retry_software_count = 2
  self.assertEquals(2, node_test_suite.retry_software_count)
  node_test_suite = test_node.delNodeTestSuite('foo')
  node_test_suite = test_node.getNodeTestSuite('foo')
  self.assertEquals(0, node_test_suite.retry_software_count)

def test_02_NodeTestSuiteWorkingDirectory(self):
  """
  Make sure we extend the working path with the node_test_suite reference
  """
  test_node = self.getTestNode()
  node_test_suite = test_node.getNodeTestSuite('foo')
  node_test_suite.edit(working_directory=self.working_directory)
  self.assertEquals("%s/foo" % self.working_directory,
                node_test_suite.working_directory)

def test_03_NodeTestSuiteCheckDataAfterEdit(self):
  """
```

```
    When a NodeTestSuite instance is edited, the method _checkData
    analyse properties and add new ones
    """
    test_node = self.getTestNode()
    node_test_suite = test_node.getNodeTestSuite('foo')
    self.updateNodeTestSuiteData(node_test_suite)
    self.assertEquals(2, len(node_test_suite.vcs_repository_list))
    repository_path_list = []
    for vcs_repository in node_test_suite.vcs_repository_list:
      repository_path_list.append(vcs_repository['repository_path'])
    expected_list = ["%s/rep0" % node_test_suite.working_directory,
                     "%s/rep1" % node_test_suite.working_directory]
    self.assertEquals(expected_list, repository_path_list)

  def test_04_constructProfile(self):
    """
    Check if the software profile is correctly generated
    """
    test_node = self.getTestNode()
    node_test_suite = test_node.getNodeTestSuite('foo')
    self.updateNodeTestSuiteData(node_test_suite, add_third_repository=True)
    test_node.constructProfile(node_test_suite)
    self.assertEquals("%s/software.cfg" % (node_test_suite.working_directory,),
                      node_test_suite.custom_profile_path)
    profile = open(node_test_suite.custom_profile_path, 'r')
    expected_profile = """
[buildout]
extends = %(temp_dir)s/testnode/foo/rep0/software.cfg

[rep1]
repository = %(temp_dir)s/testnode/foo/rep1
branch = master

[rep2]
repository = %(temp_dir)s/testnode/foo/rep2
branch = foo
""" % {'temp_dir': self._temp_dir}
    self.assertEquals(expected_profile, profile.read())
    profile.close()
```

```
def test_05_getAndUpdateFullRevisionList(self):
  """
  Check if we clone correctly repositories and get right revisions
  """
  commit_dict = self.generateTestRepositoryList()
  test_node = self.getTestNode()
  node_test_suite = test_node.getNodeTestSuite('foo')
  self.updateNodeTestSuiteData(node_test_suite)
  rev_list = test_node.getAndUpdateFullRevisionList(node_test_suite)
  self.assertEquals(2, len(rev_list))
  self.assertEquals(rev_list[0], 'rep0=2-%s' % commit_dict['rep0'][0][0])
  self.assertEquals(rev_list[1], 'rep1=2-%s' % commit_dict['rep1'][0][0])
  my_file = open(os.path.join(self.remote_repository1, 'first_file'), 'w')
  my_file.write("next_content")
  my_file.close()
  call = self.getCaller(cwd=self.remote_repository1)
  call("git commit -av -m new_commit".split() + ['--author="a b <a@b.c>"'])
  rev_list = test_node.getAndUpdateFullRevisionList(node_test_suite)
  self.assertTrue(rev_list[0].startswith('rep0=2-'))
  self.assertTrue(rev_list[1].startswith('rep1=3-'))
  self.assertEquals(2, len(node_test_suite.vcs_repository_list))
  for vcs_repository in node_test_suite.vcs_repository_list:
    self.assertTrue(os.path.exists(vcs_repository['repository_path']))

def test_06_checkRevision(self):
  """
  Check if we are able to restore older commit hash if master decide so
  """
  commit_dict = self.generateTestRepositoryList()
  test_node = self.getTestNode()
  node_test_suite = test_node.getNodeTestSuite('foo')
  self.updateNodeTestSuiteData(node_test_suite)
  rev_list = test_node.getAndUpdateFullRevisionList(node_test_suite)
  def getRepInfo(count=0, hash=0):
    assert count or hash
    info_list = []
    for vcs_repository in node_test_suite.vcs_repository_list:
      call = self.getCaller(cwd=vcs_repository['repository_path'])
      if count:
        info_list.append(
```

```
        call("git rev-list --topo-order --count HEAD".split()).strip())
      if hash:
        info_list.append(
          call("git log -n1 --format=%H".split()).strip())
    return info_list
  self.assertEquals(['2', '2'], getRepInfo(count=1))
  self.assertEquals([commit_dict['rep0'][0][0],commit_dict['rep1'][0][0]],
                    getRepInfo(hash=1))
  class TestResult(object):
    pass
  test_result = TestResult()
  # for test result to be one commit late for rep1 to force testnode to
  # reset tree to older version
  test_result.revision = 'rep0=2-%s,rep1=1-%s' % (commit_dict['rep0'][0][0],
                                        commit_dict['rep1'][1][0])
  test_node.checkRevision(test_result, node_test_suite)
  expected_count_list = ['2', '1']
  self.assertEquals(['2', '1'], getRepInfo(count=1))
  self.assertEquals([commit_dict['rep0'][0][0],commit_dict['rep1'][1][0]],
                    getRepInfo(hash=1))

def test_07_checkExistingTestSuite(self):
  test_node = self.getTestNode()
  test_suite_data = self.getTestSuiteData(add_third_repository=True)
  self.assertEquals([], os.listdir(self.working_directory))
  test_node.checkOldTestSuite(test_suite_data)
  self.assertEquals([], os.listdir(self.working_directory))
  os.mkdir(os.path.join(self.working_directory, 'foo'))
  self.assertEquals(['foo'], os.listdir(self.working_directory))
  test_node.checkOldTestSuite(test_suite_data)
  self.assertEquals(['foo'], os.listdir(self.working_directory))
  os.mkdir(os.path.join(self.working_directory, 'bar'))
  self.assertEquals(set(['bar','foo']),
                    set(os.listdir(self.working_directory)))
  test_node.checkOldTestSuite(test_suite_data)
  self.assertEquals(['foo'], os.listdir(self.working_directory))

def test_08_getSupportedParamaterSet(self):
  original_spawn = ProcessManager.spawn
  try:
```

```
    def get_help(self, *args, **kw):
      return {'stdout': """My Program
              --foo  foo
              --bar  bar"""}
    ProcessManager.spawn = get_help
    process_manager = ProcessManager(log=None)
    parameter_list = ['--foo', '--baz']
    expected_suported_parameter_set = set(['--foo'])
    supported_parameter_set = process_manager.getSupportedParameterSet(
                          "dummy_program_path", parameter_list)
    self.assertEquals(expected_suported_parameter_set, supported_parameter_set)
  finally:
    ProcessManager.spawn = original_spawn

def test_09_runTestSuite(self):
  """
  Check parameters passed to runTestSuite
  Also make sure that --firefox_bin and --xvfb_bin are passed when needed
  """
  original_getSupportedParameter = ProcessManager.getSupportedParameterSet
  original_spawn = ProcessManager.spawn
  try:
    def _createPath(path_to_create, end_path):
      os.makedirs(path_to_create)
      return os.close(os.open(os.path.join(path_to_create,
                            end_path),os.O_CREAT))
    def get_parameters(self, *args, **kw):
      call_parameter_list.append({'args': [x for x in args], 'kw':kw})
    def patch_getSupportedParameterSet(self, run_test_suite_path, parameter_list,):
     if '--firefox_bin' and '--xvfb_bin' in parameter_list:
       return set(['--firefox_bin','--xvfb_bin'])
     else:
       return []
    test_node = self.getTestNode()
    test_node.slapos_controler = SlapOSControler(self.working_directory,
                                       test_node.config)
    node_test_suite = test_node.getNodeTestSuite('foo')
    self.updateNodeTestSuiteData(node_test_suite)
    node_test_suite.revision = 'dummy'
    run_test_suite_path = _createPath(
```

```
      os.path.join(test_node.slapos_controler.instance_root,'a/bin'),'runTestSuite')
    def checkRunTestSuiteParameters(additional_parameter_list=None):
      ProcessManager.getSupportedParameterSet = patch_getSupportedParameterSet
      ProcessManager.spawn = get_parameters
      test_node.runTestSuite(node_test_suite,"http://foo.bar")
      expected_parameter_list = ['%s/a/bin/runTestSuite'
          %(test_node.slapos_controler.instance_root), '--test_suite', 'Foo', '--revision',
          'dummy', '--test_suite_title', 'Foo-Test', '--node_quantity', 3, '--master_url',
          'http://foo.bar']
      if additional_parameter_list:
        expected_parameter_list.extend(additional_parameter_list)
      self.assertEqual(call_parameter_list[0]['args'], expected_parameter_list)
    call_parameter_list = []
    checkRunTestSuiteParameters()
    _createPath(os.path.join(test_node.config['slapos_directory'], 'soft/a/parts/firefox'),'firefox-slapos')
    _createPath(os.path.join(test_node.config['slapos_directory'], 'soft/a/parts/xserver/bin'),'Xvfb')
    call_parameter_list = []
    checkRunTestSuiteParameters(additional_parameter_list=['--firefox_bin',
      '%s/soft/a/parts/firefox/firefox-slapos'
       %(test_node.config['slapos_directory']),
      '--xvfb_bin',
      '%s/soft/a/parts/xserver/bin/Xvfb'
        %(test_node.config['slapos_directory'])])
  finally:
    ProcessManager.getSupportedParameterSet   = original_getSupportedParameter
    ProcessManager.spawn = original_spawn

def test_10_prepareSlapOS(self):
  test_node = self.getTestNode()
  test_node_slapos = SlapOSInstance()
  node_test_suite = test_node.getNodeTestSuite('foo')
  node_test_suite.edit(working_directory=self.working_directory)
  status_dict = {"status_code" : 0}
  global call_list
  call_list = []
  class Patch:
    def __init__(self, method_name, status_code=0):
      self.method_name = method_name
      self.status_code = status_code
    def __call__(self, *args, **kw):
```

```
      global call_list
      call_list.append({"method_name": self.method_name,
                        "args": [x for x in args],
                        "kw": kw})
      return {"status_code": self.status_code}
    SlapOSControler.initializeSlapOSControler = Patch("initializeSlapOSControler")
    SlapOSControler.runSoftwareRelease = Patch("runSoftwareRelease")
    SlapOSControler.runComputerPartition = Patch("runComputerPartition")
    test_node.prepareSlapOSForTestNode(test_node_slapos)
    self.assertEquals(["initializeSlapOSControler", "runSoftwareRelease"],
                      [x["method_name"] for x in call_list])
    call_list = []
    test_node.prepareSlapOSForTestSuite(node_test_suite)
    self.assertEquals(["initializeSlapOSControler", "runSoftwareRelease",
                       "runComputerPartition"],
                      [x["method_name"] for x in call_list])
    call_list = []
    SlapOSControler.runSoftwareRelease = Patch("runSoftwareRelease", status_code=1)
    self.assertRaises(SubprocessError, test_node.prepareSlapOSForTestSuite,
                      node_test_suite)

  def test_11_run(self):
    def doNothing(self, *args, **kw):
      pass
    test_self = self
    test_result_path_root = os.path.join(test_self._temp_dir,'test/results')
    os.makedirs(test_result_path_root)
    global counter
    counter = 0
    def patch_startTestSuite(self,test_node_title):
      global counter
      config_list = []
      def _checkExistingTestSuite(reference_set):
        test_self.assertEquals(set(reference_set),
                               set(os.listdir(test_node.config["working_directory"])))
        for x in reference_set:
          test_self.assertTrue(os.path.exists(os.path.join(
                                 test_node.config["working_directory"],x)),True)
      if counter == 0:
        config_list.append(test_self.getTestSuiteData(reference='foo')[0])
```

```
    config_list.append(test_self.getTestSuiteData(reference='bar')[0])
  elif counter == 1:
    _checkExistingTestSuite(set(['foo']))
    config_list.append(test_self.getTestSuiteData(reference='bar')[0])
    config_list.append(test_self.getTestSuiteData(reference='foo')[0])
  elif counter == 2:
    _checkExistingTestSuite(set(['foo','bar']))
    config_list.append(test_self.getTestSuiteData(reference='foo')[0])
    config_list.append(test_self.getTestSuiteData(reference='qux')[0])
  elif counter == 3:
    _checkExistingTestSuite(set(['foo','qux']))
    config_list.append(test_self.getTestSuiteData(reference='foox')[0])
  elif counter == 4:
    _checkExistingTestSuite(set(['foox']))
    config_list.append(test_self.getTestSuiteData(reference='bax')[0])
  elif counter == 5:
    _checkExistingTestSuite(set(['bax']))
    raise StopIteration
  counter += 1
  return json.dumps(config_list)
def patch_createTestResult(self, revision, test_name_list, node_title,
      allow_restart=False, test_title=None, project_title=None):
  global counter
  # return no test to check if run method will run the next test suite
  if counter == 3 and project_title != 'qux':
    result = None
  else:
    test_result_path = os.path.join(test_result_path_root, test_title)
    result =  TestResultProxy(self._proxy, self._retry_time,
          self._logger, test_result_path, node_title, revision)
  return result
original_sleep = time.sleep
time.sleep = doNothing
self.generateTestRepositoryList()
original_startTestSuite = TaskDistributor.startTestSuite
TaskDistributor.startTestSuite = patch_startTestSuite
original_createTestResult = TaskDistributionTool.createTestResult
TaskDistributionTool.createTestResult = patch_createTestResult
test_node = self.getTestNode()
original_prepareSlapOS = test_node._prepareSlapOS
```

```
  test_node._prepareSlapOS = doNothing
  original_runTestSuite = test_node.runTestSuite
  test_node.runTestSuite = doNothing
  SlapOSControler.initializeSlapOSControler = doNothing
  try:
   test_node.run()
  except Exception as e:
   self.assertEqual(type(e),StopIteration)
  finally:
   time.sleep = original_sleep
   TaskDistributor.startTestSuite = original_startTestSuite
   TaskDistributionTool.createTestResult = original_createTestResult
   test_node._prepareSlapOS = original_prepareSlapOS
   test_node.runTestSuite = original_runTestSuite

 def test_12_spawn(self):
  def _checkCorrectStatus(expected_status,*args):
   result = process_manager.spawn(*args)
   self.assertEqual(result['status_code'], expected_status)
  process_manager = ProcessManager(log=self.log, max_timeout=1)
  _checkCorrectStatus(0, *['sleep','0'])
  _checkCorrectStatus(-15, *['sleep','2'])
```

Exemple d'un algorithme Pixastic implémenté pour SafeImage

```
var noise = function(dataI,width,height) {

   var imagedata = dataI;
   var data = imagedata.data;
   var w = width;
   var h = height;
   var mode = 1;
                  var w4 = w*4;
                  var y = h;


                        do {
                              var offsetY = (y-1)*w4;

                              var nextY = (y == h) ? y - 1 : y;
```

```
var prevY = (y == 1) ? 0 : y-2;

var offsetYPrev = prevY*w*4;
var offsetYNext = nextY*w*4;

var x = w;
do {
        var offset = offsetY + (x*4-4);

        var offsetPrev = offsetYPrev + ((x == 1) ? 0 : x-2) * 4;
        var offsetNext = offsetYNext + ((x == w) ? x-1 : x) * 4;

        var minR, maxR, minG, maxG, minB, maxB;

        minR = maxR = data[offsetPrev];
        var r1 = data[offset-4], r2 = data[offset+4], r3 = data[offsetNext];
        if (r1 < minR) minR = r1;
        if (r2 < minR) minR = r2;
        if (r3 < minR) minR = r3;
        if (r1 > maxR) maxR = r1;
        if (r2 > maxR) maxR = r2;
        if (r3 > maxR) maxR = r3;

        minG = maxG = data[offsetPrev+1];
        var g1 = data[offset-3], g2 = data[offset+5], g3 = data[offsetNext+1];
        if (g1 < minG) minG = g1;
        if (g2 < minG) minG = g2;
        if (g3 < minG) minG = g3;
        if (g1 > maxG) maxG = g1;
        if (g2 > maxG) maxG = g2;
        if (g3 > maxG) maxG = g3;

        minB = maxB = data[offsetPrev+2];
        var b1 = data[offset-2], b2 = data[offset+6], b3 = data[offsetNext+2];
        if (b1 < minB) minB = b1;
        if (b2 < minB) minB = b2;
        if (b3 < minB) minB = b3;
        if (b1 > maxB) maxB = b1;
        if (b2 > maxB) maxB = b2;
        if (b3 > maxB) maxB = b3;
```

```
                        if (data[offset] > maxR) {
                                data[offset] = maxR;
                        } else if (data[offset] < minR) {
                                data[offset] = minR;
                        }
                        if (data[offset+1] > maxG) {
                                data[offset+1] = maxG;
                        } else if (data[offset+1] < minG) {
                                data[offset+1] = minG;
                        }
                        if (data[offset+2] > maxB) {
                                data[offset+2] = maxB;
                        } else if (data[offset+2] < minB) {
                                data[offset+2] = minB;
                        }

                } while (--x);
        } while (--y);

    imagedata.data = data;
                return imagedata;

}
```