Cahier des charges

Corentin GUILLEVIC et Sarah HADBI

March 20, 2015

1 Contexte et définition du problème

De nos jours, toute personne disposant d'un ordinateur a besoin d'y installer un certain nombre de logiciels pour l'utiliser à des fins personnelles ou professionnelles. Sachant que les ordinateurs fonctionnent sous différentes architectures (32 bits, 64 bits, ARM...) et systèmes d'exploitations (Windows, MAC OS X, Linux...), eux-mêmes divisés dans le cas de Linux en distributions (Debian , Red hat...), à partir de là un problème se pose :

Si on prend l'exemple de Linux, un logiciel est installé à travers un paquet récupéré sur le dépôt d'une distribution. Le paquet en question n'est pas forcément disponible sous toutes les distributions. A partir de là, si une personne possède une distribution spécifique sur son ordinateur, et que le paquet du logiciel n'est pas forcément adapté pour celle-ci (c'est-à-dire qu'il n'existe pas pour la distribution), elle ne pourra pas l'installer directement et simplement sur sa machine. Elle sera donc contrainte d'aller récupérer les sources du paquet et de devoir assurer la compilation, tâche assez laborieuse.

2 Projet Paquito

2.1 Existant

Pour répondre à ce besoin de simplification, le projet Paquito a été initié, qui dispose actuellement d'une version préalable composée d'un certain nombre de scripts. Ceux-ci visent à la création assistée de paquets à destination de la distribution Debian. Pour la poursuite du projet, ces scripts peuvent éventuellement servir de base.

L'objectif est de faciliter d'une part la vie des développeurs dans la mise à disposition de leurs logiciels et d'autre part faciliter l'accès aux logiciels

pour les utilisateurs. Le projet Paquito se matérialise par une infrastructure de construction de paquet à destination des distributions Linux.

Ce projet doit offrir à terme une compilation automatique et simultanée de paquets, à partir d'un dépôt quelconque mis à disposition par le développeur. Selon les choix du développeur, un ensemble de paquets pour différentes distributions, architectures et versions sera généré. Cet ensemble sera réparti dans les différents dépôts associés.

L'infrastructure Paquito doit aussi fournir au développeur un retour en cas d'erreurs ainsi qu'un moyen de vérifier et tester les résultats produits (avant de les publier).

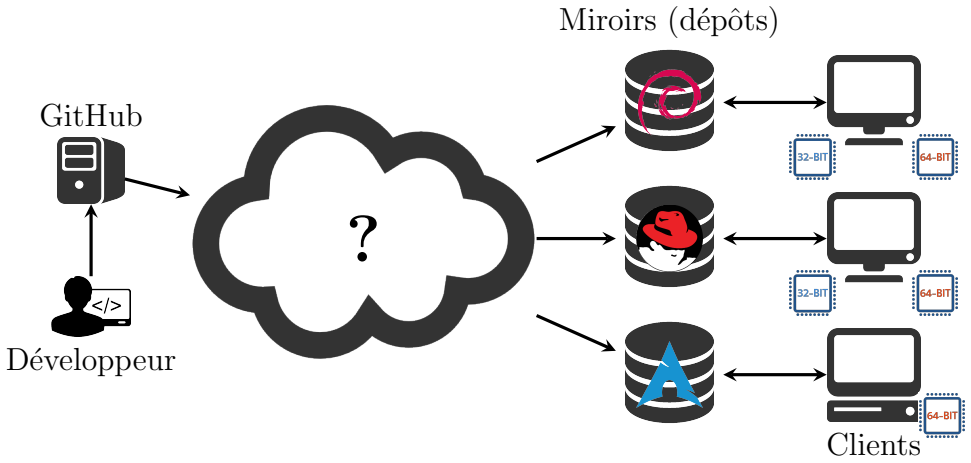

Figure 1: Illustration du problème

2.2 Intervenants

Développeurs :

- Corentin GUILLEVIC
- Sarah HADBI

Clients :

- Francis HULIN-HUBARD
- Alban LINARD
- Pierre-Arnaud SENTUCQ

2.3 Objectifs

L'objectif initial du projet Paquito (qui existe depuis 2014) était de faciliter la conception de paquets pour la distribution Debian. Dans le cadre de ce PSAR, le but est de reprendre l'existant et de l'améliorer afin d'étendre ses fonctionnalités de génération de paquets, pour les distributions :

- Archlinux
- Redhat

Sachant que l'architecture doit être prise en compte pour chacune de ces distributions (32 et 64 bits, sauf pour Archlinux qui n'acceptera que le 64 bits).

3 Définitions

- *Paquet* : Archive (fichier compressé) comprenant les fichiers informatiques, les informations et procédures nécessaires à l'installation d'un logiciel sur un système d'exploitation .
- *Compilation* : Travail réalisé par un compilateur qui consiste à transformer un code source lisible par un humain en un fichier binaire exécutable par une machine.
- *Dépôt/Miroir de paquets* : Un dépôt est un stockage centralisé et organisé de paquets, en vue de leur distribution sur le réseau ou bien un endroit directement accessible aux utilisateurs.
- *Docker* : Logiciel automatisant le déploiement d'applications dans des conteneurs logiciels et empaquette une application et ses dépendances dans un conteneur virtuel, qui pourra être exécuté sur n'importe quel serveur Linux . Ceci permet d'étendre la flexibilité et la portabilité d'exécution d'une application, que ce soit sur la machine locale, un cloud privé ou public, une machine nue.
- *Container* : Un type de cloisonnement d'un système d'exploitation dans certains systèmes de virtualisation légers réutilisant le noyau et éventuellement les bibliothèques du système hôte.
- *GitHub* : Service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git , Le nom GitHub est composé du mot "git" faisant référence à un système de

contrôle de version open-source et le mot "hub" faisant référence au réseau social bâti autour du système Git.

- *Jenkins* : Outil d'intégration continue, il s'interface avec des systèmes de gestion de versions tels que CVS, Git et Subversion.
- *Dépendances* : un logiciel dépend pour son exécution de la présence (inclusion) des bibliothèques logicielles adéquates.

4 Description fonctionnelle

L'infrastructure utilisée dans le cadre du projet Paquito est illustrée par le schéma suivant :

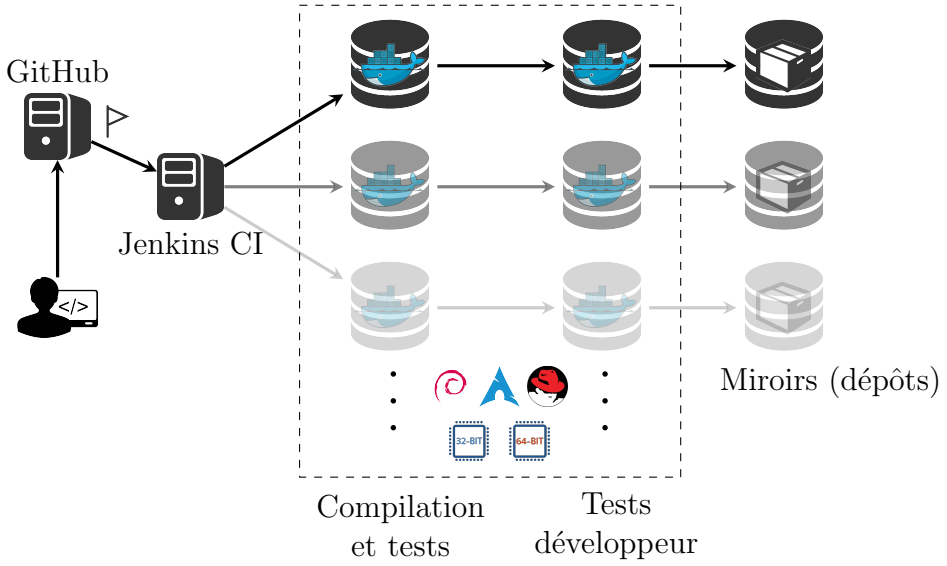

Figure 2: Illustration de la solution

La chaîne de fonctionnement de l'infrastructure est divisée en plusieurs étapes :

- Un développeur effectue un commit dans le dépôt GitHub (c'est-à-dire qu'il officialise dans le dépôt les modifications qu'il a apporté au code source). L'interface GitHub, qui offre des services administratifs, active un fanion indiquant que des changements ont récemment eu lieu.

  Remarque : Parmi le code source que le développeur a officialisé se trouve un fichier de configuration. Ce fichier, primordial pour l'infrastructure Paquito, contient un ensemble de méta-données telles que le nom du paquet ou sa version, mais les plus importantes sont les dépendances.

- Le service Jenkins CI, en veille de l'activation du fanion du dépôt GitHub, détecte un changement et procède donc au téléchargement du projet (il récupère par la même occasion un maximum d'informations qui serviront lors de la phase de compilation). Il lancera ensuite une série de containers Docker adapté à chaque distribution, ses versions et les architectures.

  Remarque : Normalement utilisé pour réaliser de l'intégration continue, Jenkins CI est utilisé dans le cadre du projet uniquement pour ses fonctionnalités d'écoute de dépôts et de démarrage de containers Docker.

- Chaque container Docker (qui a une configuration vierge) se chargera de compiler le code source (notamment à partir des informations récoltées par Jenkins CI) en considérant les dépendances (autre logiciel, librairie...). En cas d'échec, le container concerné renvoi les messages d'erreurs au développeur puis s'arrête (les autres containers ne sont pas impactés).

  Remarque : Les éventuelles erreurs renvoyées par le container seront sauvegardées sous forme d'un fichier de journalisation.

- Une fois la compilation réussie et le package créé, celui-ci est testé sur un nouveau container Docker créé à cet effet (comme pour la compilation, ce container est vierge). L'idée est de s'assurer que l'installation du package aboutisse avec succès et que le programme associé fonctionne correctement. Ce dernier point est vérifié grâce aux tests arbitraires que le développeur fournit sous formes de scripts . En cas d'échec d'un test, le container en question est arrêté et le développeur est notifié.

- Une fois que le package est réputé conforme et fonctionnel, celui-ci est inséré dans un dépôt de logiciel lié à sa distribution, sa version et

son architecture. Les utilisateurs n'auront qu'à télécharger le logiciel par l'intermédiaire de leur gestionnaire de paquets (après avoir bien entendu ajouté l'adresse du miroir où se trouve le package dans le fichier **source.list**).

5 Scénario de recette

Le scénario imaginé pour valider l'avancement du présent projet sera le suivant : à partir de 3 projets mis à disposition par les clients (*Hello World*, *Prog* et *GrepSTN*), nous devons générer un ensemble de paquets pour les distributions Debian , Redhat et ArchLinux (en prenant en compte les différentes architectures, c'est-à-dire 32bits et 64bits). Plus précisément, la validation de ce scénario passera par les points suivants :

- Identifier les points communs et les différences pour la construction des paquets (selon les systèmes de gestions de paquets de chaque distribution)
- Modifier le fichier de configuration du projet Paquito 2014 pour prendre en compte les nouveaux types de paquets (en se basant notamment sur les recherches sur les points communs et les différences entre types de paquets) ainsi que pour y inclure toutes les dépendances nécessaires pour la génération de ces paquets.
- Modifier et adapter si nécessaire les scripts du projet Paquito 2014 pour ajouter le support des paquets Redhat et Archlinux, en plus des paquets Debian.

La condition de validation finale de ce scénario est donc d'obtenir un ensemble de paquets installables et fonctionnels (les logiciels doivent fonctionner) des projets *Hello World*, *Prog* et *GrepSTN* pour les distributions cibles.

6 Délais

Définition des dates-clés pour la remise des livrables :
Semaine du 06/03 : STBE.
Semaine du 06/04 : STR.
Semaine du 05/05 : Soutenance.