Outils libres pour la gestion d'un parc macOS

Mickael Masquelin

Outils libres pour la gestion d'un parc macOS

Mickaël Masquelin
Institut d'Electronique, de Microélectronique et de Nanotechnologie (IEMN – UMR8520)
Avenue Henri Poincaré
CS60069
59652 Villeneuve d'Ascq Cedex

Résumé

Auparavant, les ordinateurs sous MacOs étaient plutôt minoritaires dans nos unités de recherche. La tendance s'est inversée et ces machines sont désormais omniprésentes dans nos réseaux locaux très hétérogènes.

Même s'ils sont pourvus de solides fonctionnalités de sécurité, les systèmes d'exploitation Apple peuvent être aussi vulnérables que ceux édités par Microsoft. Leur gestion peut alors rapidement devenir un véritable casse-tête, surtout après avoir livré la machine à l'utilisateur. Il est donc nécessaire de les intégrer dans une gestion globale de parc informatique.

Les solutions techniques dédiées à leur administration système ont beaucoup évolué : les catalogues de « golden master images » ont laissé leur place à de nouveaux outils, libres. Ce poster vise à les présenter et montrer leurs interactions.

BSDPy, associé à AutoNBI et Imagr, rend le processus de déploiement plus « dynamique ». Ces applications permettent de construire et dérouler un scénario à l'issue de l'installation de base du système, en fonction du profil de l'utilisateur (via des protocoles support comme HTTP).

D'autres logiciels, tels que le couple Munki/Autopkg, permettent de gérer un catalogue d'applications en self-service, standardisé et centralisé, à l'aide d'un simple serveur web. Il est très simple d'utilisation car il s'inspire de l'App Store. Le reporting est, quant à lui, assuré par la solution Munkireport.

Grâce à l'intégration avec le programme DEP (Device Enrollment Program) d'Apple, les administrateurs peuvent permettre l'installation automatique des profils de MDM (comme MicroMDM). Ces solutions libres montrent qu'il est ainsi possible de déployer des applications par profils pour le poste de travail MacOs.

Mots-clefs

MacOs, package, déploiement, applications, catalogue, self-service, Apple, automatisation, mises à jour, logiciel libre, gestion de parc, centralisation

1 Introduction

Il y a quelques années, les machines *Apple* représentaient à peine 5% du parc informatique de notre laboratoire contre environ 20% aujourd'hui : nous recensons désormais approximativement 1600 objets connectés au réseau local ou au réseau sans fil pour environ 450 utilisateurs. Le passage à une architecture *Intel* et l'engouement suscité par ces environnements permettent d'expliquer, en partie, le phénomène observé.

Au fil du temps, la courbe d'adoption des ordinateurs exécutant *MacOs* au sein de notre contexte n'a fait que croitre. Très vite, nous nous sommes heurtés à des problèmes relatifs à la gestion des logiciels, des mises à jour (aussi bien pour le système d'exploitation que pour les applications) et des politiques de sécurité.

La solution que nous avons retenue s'appuie essentiellement sur des outils libres. Les raisons qui ont motivé ces choix sont multiples. Tout d'abord, le coût financier était « limité »: il n'était pas nécessaire de déployer une infrastructure système spécifique pour gérer cet environnement (systèmes serveurs *Apple MacOs Server* et compléments de gestion commerciaux tels que *Jamf Pro, Casper Suite, VMware AirWatch*, ...).

Par ailleurs, la solution technique était maitrisée de bout en bout car implémentée et déployée par nos soins (pas de recours à un service en ligne, dans le « *Cloud* »). Enfin, pouvoir s'appuyer sur une grande communauté d'utilisateurs était également un critère très important dans le choix des solutions techniques: elle est certes anglophone, mais très à l'écoute et très réactive.

Nous avons donc fait le choix de mettre en œuvre progressivement un lot de briques logicielles qui s'appuient sur le réseau local du laboratoire et un serveur web (*Apache 2*). Cet ensemble s'est vu enrichi au fil du temps, en fonction des besoins des utilisateurs mais aussi grâce aux évolutions des différentes solutions déployées.

2 Munki

Nous avons tout d'abord déployé le service *open source Munki* (publié sous les termes de la licence *Apache* en version 2.0). Cet outil a été développé par les équipes techniques des studios d'animation *Walt Disney* pour la gestion de leur parc informatique. Il est dédié à l'écosystème *MacOs* et a été initialement publié en 2011.

Son rôle principal consiste à installer les logiciels demandés par le biais d'une interface, personnalisable, que les utilisateurs prennent facilement en main car elle ressemble à *l'App Store*. Autre atout non négligeable : il n'est pas nécessaire de disposer des droits d'administrateur pour être exécuté côté client.

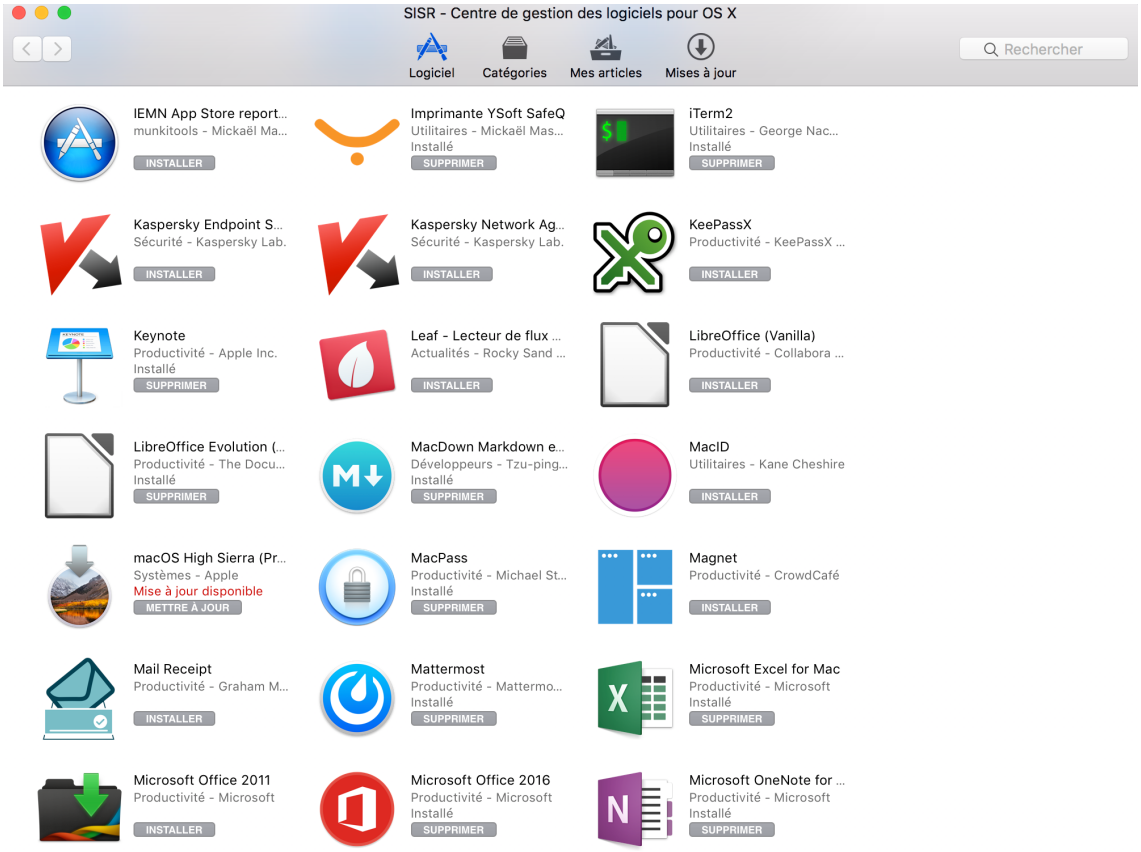

Figure 1 - Centre de gestion des logiciels pour MacOs, déployé lors de l'installation de Munki

L'application prend en charge les programmes d'installation de logiciels au format de paquet *Apple* (type *.PKG*), les paquets dédiés au déploiement pour l'entreprise (comme ceux *d'Adobe CS5/CS6* …) et les images disques comme sources d'installation (type *.DMG*). Il permet aussi le déploiement d'imprimantes.

Elle permet ainsi de créer et de mettre à disposition de nos utilisateurs un catalogue logiciel standardisé et centralisé. Elle offre également la possibilité de publier automatiquement (ou sur demande) des applications mais aussi des mises à jour du système et de spécifier l'ordre de leur exécution.

Pour organiser l'opération de déploiement de ces logiciels sur les machines clientes, l'administrateur système pourra s'appuyer sur des « manifestes » qu'il aura rédigés.

Un « manifeste » est le terme consacré pour désigner un fichier « texte » structuré avec un formalisme *XML*. Ce fichier définit la liste des logiciels qu'il faut installer, mettre à jour (obligatoirement ou non), supprimer ou vérifier leur présence effective sur l'hôte cible. Par ailleurs, un « manifeste » peut lui-même faire référence à un autre « manifeste ».

Par exemple, on peut créer un « manifeste » pour installer certains logiciels dédiés à un groupe de personnes … qui fait lui-même appel à un « manifeste » incluant des applications communes, déployées pour tout le laboratoire (à l'image de logiciels comme *Mozilla Firefox* ou *Mozilla Thunderbird*, plutôt des standards).

Les commandes et outils classiques de manipulation de fichiers « texte » (*grep/sed/awk/vim*) peuvent être utilisés pour modifier son contenu.

Un agent, déployé au préalable sur les clients (*Munki Tools*), permet la communication avec le serveur web *Apache 2* via le protocole *HTTPS*. Il n'y a pas de *CGI* côté serveur, mais une arborescence spécifique qui contient 3 choses :

- Les applications mises à disposition des utilisateurs ;
- Des catalogues de logiciels créés à partir de la liste des logiciels disponibles ;
- Des « manifestes » décrivant les opérations à effectuer sur les logiciels.

Les applications sont donc chargées sur le serveur à l'aide d'une commande interne dans Munki (*munkiimport*).

3 Reposado

Munki peut également intervenir pour gérer les mises à jour des systèmes d'exploitation *MacOs*. En effet, il est possible d'indiquer à l'application l'adresse d'un dépôt local qui contient les mises à jour et derniers correctifs publiés par *Apple* pour ses plateformes.

Comment maitriser cette partie du processus ? C'est encore une fois les équipes techniques des studios d'animation *Walt Disney* qui sont à l'origine d'un ensemble d'outils libres écrits en *Python* (sous licence BSD), appelés *Reposado*.

Ces outils autorisent une gestion plus fine des mises à jour sur les machines avec la possibilité de créer un serveur de mises à jour local. Celui-ci aura pour rôle de diffuser, via le protocole *HTTPS*, cet ensemble de correctifs et/ou patches vers les machines des utilisateurs, en fonction des besoins.

Cette suite de logiciels est dédiée uniquement à ces opérations. Elle ne permet pas la distribution des applications.

Le processus est relativement simple :

- Le programme « *repo_sync* » récupère depuis les serveurs *Apple* le catalogue des mises à jour logicielles disponibles pour les systèmes *MacOs* ;
- Si l'administrateur système le souhaite, cette application peut également télécharger les correctifs afin de les servir à ses clients locaux via un serveur web (le serveur utilisé pour distribuer les logiciels avec *Munki* peut être utilisé pour cette tâche).

(Cela rappelle le fonctionnement de *Windows Server Update Services* utilisé pour les environnements *Microsoft)*.

En plus de permettre une économie non négligeable de bande passante sur le site, il est ainsi possible de mettre en place des branches de test des mises à jour sur un nombre limité de machines (virtuelles par exemple) avant de les pousser sur le parc en production.

4 Munkireport-php

La plupart du temps, le processus de déploiement ou de mise à jour des clients fonctionne relativement bien. Néanmoins, il peut être utile pour l'administrateur système d'avoir une vue, en temps réel, sur la « conformité » ou non du parc informatique en production.

Munkireport-php nous a apporté une réponse efficace à cette problématique. Il s'agit d'une interface web conçue pour effectuer des opérations de *reporting* depuis les clients *Munki*. Ecrite en *PHP* et publiée sous les termes de la licence MIT, elle a besoin d'un serveur web en support pour être pleinement fonctionnelle.

Cette application affiche de manière synthétique les erreurs, les installations qui sont en attente d'exécution sur les clients ou encore d'autres indicateurs sur les systèmes hôtes (espace disque encore disponible sur les machines, *uptime* etc …).

5 AutoPkg

Compte tenu de la diversité des logiciels utilisés par notre communauté d'utilisateurs, la tâche de maintenance du catalogue de logiciels a été rendue très vite complexe. Nous avons pu répondre à cette problématique à l'aide d'un autre couple d'applications : *AutoPkg* et son interface graphique *AutoPkgr* (éditée par *Linde Group*). Ces deux applications sont publiées sous les termes de la licence *Apache 2.0*.

AutoPkg permet d'interroger les services de mise à jour des différents logiciels pour trouver les dernières versions des applications et les pousser ainsi vers le dépôt *Munki*. Il permet donc l'automatisation de la gestion de ces mises à niveau de logiciels.

La communauté des administrateurs de solution *MacOs* joue ici un rôle assez important car elle a construit une base partagée de fichiers appelées « recettes » (plusieurs dépôts sont disponibles sur *Github*). Ces « recettes » spécifient l'endroit où télécharger les applications et quelques actions complémentaires à effectuer éventuellement avant de les rendre exploitables par *Munki*.

Par exemple, lors du processus de rafraîchissement du contenu du catalogue, *AutoPkg* va venir lire les contenus des flux « *Sparkle* » (un *framework* très utilisé par les développeurs pour gérer le processus de publication des logiciels) sur les différents sites hébergeant ces différents outils. Vous pouvez tout à fait écrire votre propre « recette » mais aussi venir enrichir le contenu d'une « recette » existante ou l'adapter pour inclure vos spécificités locales : on effectue alors une surcharge des réglages d'*AutoPkg*.

Des fonctions avancées dans *AutoPkgr* permettent d'aller encore plus loin dans le processus. Ainsi, il est possible de configurer un ensemble de tâches qui seront exécutées périodiquement. De cette façon, la liste des dépôts et les applications associées à ces derniers peut être facilement rafraichies sur le serveur. L'administrateur système peut également recevoir un courriel récapitulatif, pratique pour éditer un journal des changements locaux.

Seul bémol dans ce fonctionnement : l'utilisation des « recettes » proposées par la communauté implique de faire confiance à des tiers pour la récupération et l'installation des applications sur vos postes clients. Des mécanismes de contrôle existent toutefois (via le contrôle des sommes de *hashage* des fichiers), mais il est nécessaire d'intégrer ce risque lors du déploiement.

6 BSDPy, AutoNBI et Imagr

La chaîne serait presque complète et nous couvrons de cette façon pratiquement tous les cas de figure ... mais que se passe-t-il par exemple lorsque l'utilisateur restitue son poste de travail et qu'il est réattribué, que le disque dur de celui-ci tombe en panne et qu'il soit nécessaire, après son remplacement, de réinstaller le système ? Il devient alors nécessaire de reprendre toutes les étapes précédentes et de dérouler toutes les étapes de l'installation, à la main.

Nous avons cherché et proposé là encore, une réponse efficace et complémentaire à nos utilisateurs pour limiter les contraintes induites par ce processus laborieux. Elle repose encore une fois sur le réseau local et un serveur web pour répondre à cette problématique.

Ainsi, nous avons installé le couple *BSDPy* (publié sous les termes de la licence Apache 2.0), associé à un outil capable de gérer des images des systèmes au format *NetBoot (NBI)* appelé *AutoNBI* (écrit en *Python*). Ceux-ci permettent de se substituer efficacement à un serveur *Apple NetBoot*.

Les outils permettent à l'utilisateur d'initier le processus d'installation ou de réinstallation par le réseau (un peu à la manière d'un déploiement via *PXE*). De cette façon, il est assez rapide de rendre une machine fonctionnelle à l'utilisateur. En effet, *Apple* permet la réalisation de master du système (« *golden images* »), qui peuvent être déployées sur les clients indépendamment de la machine (*MacBook, iMac, Mac Pro, ...*).

Cependant, avec les dernières versions des systèmes d'exploitation *MacOs* (et surtout depuis 10.13), ce processus est devenu caduc. Nous avons néanmoins trouvé une parade qui consiste, encore une fois, à recourir à un logiciel libre pour répondre à cette problématique.

Aussi, l'application *Imagr* (publiée sous les termes de la licence *Apache 2.0*), développée par Graham Gilbert en *Python*, permet de réaliser le processus de déploiement via le protocole HTTPS, d'un package d'installation de *MacOs* depuis le réseau (à la manière du système de récupération par le réseau développé par *Apple*).

L'outil ne se contente pas seulement de permettre l'installation: il autorise, à l'issue de l'opération, l'exécution de scripts ou l'installation de packages complémentaires en vue de compléter le processus côté client.

Cette étape nous permet d'installer automatiquement les *Munki Tool*s sur les postes clients et de les configurer avec les paramètres du serveur web *Munki* dès l'installation initiale du système. La chaîne est désormais presque complète … !

7 MicroMDM

Depuis quelques années, Apple propose une solution qui aide les entreprises et autres structures à déployer et configurer facilement des appareils sous *MacOS* et *iOS*. Elle repose sur l'utilisation d'une solution capable de gérer les appareils mobiles (*Mobile Device Management* ou *MDM*) et est appelée le « Programme d'inscription des appareils » (*Device Enrollment Program* ou *DEP*).

En venant lier la solution *MDM* gérée localement avec les serveurs d'*Apple* prenant en charge le « Programme d'inscription des appareils », les clients gérés reçoivent automatiquement un certain nombre de préréglages lors de leur première initialisation (via une liaison Internet).

Nous expérimentons ainsi une solution libre, appelée *MicroMDM*, qui permet de gérer des terminaux *Apple MacOs*. C'est encore un projet *open source*, écrit en *Go* (publié sous les termes de la licence MIT), relativement jeune, mais prometteur. Ce serveur *HTTP* implémente une partie des spécifications *MDM* exposées par *Apple*. Le projet se concentre pour le moment sur la partie *DEP* et le provisionnement des terminaux *Apple*. Il est pilotable par le biais d'une *Application Programming* Interface (*API*).

A court/moyen terme, cette étape supplémentaire dans le processus nous permettra de réaliser l'installation initiale d'une application (en l'occurrence *Munki*) et d'optimiser, de cette façon, la gestion

globale du cycle de vie des postes de travail sous *MacOs* de notre laboratoire (réduisant l'empreinte du processus d'installation initiale/réinstallation du système d'exploitation).

8 Conclusion

Cet article montre, dans les grandes lignes, toute la chaîne et les briques logicielles libres qui peuvent être déployées pour gérer un parc de machines sous *MacOs*. Elles présentent de nombreux atouts, comme le fait de pouvoir automatiser certaines tâches, d'autoriser le déploiement à la demande de l'utilisateur, de proposer une interface de gestion très conviviale pour l'usager final, etc…

Même si elles nécessitent un peu d'« huile de coude » pour leur mise en œuvre, le « retour sur investissement » est très rapide au-delà de 5 postes et ces outils nous paraissent désormais indispensables pour une gestion de parc informatique maitrisée et sécurisée.