Être administrateur systèmes : ne pas s'enfermer dans une spécialité ? (Libres conseils 8/42)

Chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang :lerouge, lamessen, CoudCoud, Kev, peupleLa (relectures),Goofy, Jej, Julius22, kalupa, 4nti7rust, ga3lig, Tsigorf, maat

Aimer l'inconnu

Jeff Mitchell

Jeff Mitchell passe ses journées de travail à s'activer sur tout ce qui touche aux ordinateurs et aux réseaux et son temps libre à barboter dans toutes sortes de projets de logiciels libres et open source. Ce qu'il préfère c'est la convergence des deux. Après avoir travaillé en tant qu'administrateur systèmes professionnel de 1999 à 2005, il maintient son niveau de compétences en les mettant bénévolement au service de projets libres en divers lieux. Ces temps-ci, son activité pour le Libre est dédiée à l'administration systèmes pour KDE[1] et c'est l'un des développeurs principaux du lecteur Tomahawk[2]. Jeff vit actuellement à Boston, aux États-Unis.

Récemment, à mon travail, j'ai fait partie d'une équipe qui faisait passer les entretiens d'embauche pour un poste d'administrateur systèmes. Après avoir parcouru quelques dizaines de curriculum vitae nous avons finalement convoqué notre premier candidat. Celui-ci — appelons-le John — avait aussi bien l'expérience de petites structures, style laboratoire informatique, que de plus vastes opérations dans des centres de données. À première vue, les choses se présentaient bien, si ce n'est qu'il avait eu cette réponse bizarre à quelques-unes de nos questions : « je suis administrateur systèmes ». Le sens de cette phrase n'a pas été immédiatement clair pour nous, jusqu'à ce que l'échange suivant ait lieu :

> *Moi : Donc, vous avez dit que vous n'avez pas d'expérience avec Cisco IOS,*

mais qu'en est-il des réseaux en général ?

John : Eh bien, je suis administrateur systèmes.

Moi : Oui, mais que diriez-vous sur les concepts de réseau ? Les protocoles de routage comme BGP ou OSPF, les VLANs, les ponts réseaux…

John, exaspéré : Je suis administrateur systèmes.

C'est à ce moment-là que nous avons compris ce qu'il voulait dire. John ne nous disait pas qu'il connaissait toutes ces choses que nous lui demandions puisqu'il était administrateur systèmes ; il nous expliquait que *parce qu*'il était administrateur systèmes, il n'en savait rien. John était administrateur systèmes et cela signifiait pour lui que ces tâches étaient celles d'un administrateur réseau. Sans surprise, John n'a pas obtenu le poste.

Dans bien des projets *open source*, la spécialisation est une malédiction et non une bénédiction. Qu'un projet relève d'une catégorie ou l'autre dépend souvent de la taille de l'équipe de développement ; la spécialisation à l'extrême peut entraîner de graves perturbations dans un projet en cas de départ d'un développeur, que ce soit en bons ou mauvais termes, qu'on le regrette ou non. Il en va de même pour les administrateurs systèmes de projets *open source*, bien que la pénurie générale de ces derniers semble autoriser aux projets une marge de tolérance parfois dangereuse.

L'exemple le plus flagrant qui me vienne à l'esprit impliquait un projet spécifique dont le site de documentation (y compris toute celle de l'installation et de la configuration) était indisponible depuis plus d'un mois. La raison : le serveur était

en panne et la seule personne qui en avait l'accès naviguait sur un « bateau pirate » avec les membres du parti pirate suédois. C'est une histoire vraie.

Cependant, tous les points de défaillance ne sont pas dus à l'absence des administrateurs systèmes ; certains sont artificiels. Sur un gros projet, les décisions des droits d'accès à l'administration systèmes étaient assumées par un seul administrateur. Il ne s'était pas seulement réservé certains droits d'accès uniquement pour lui-même (vous l'avez deviné : oui, il a disparu pendant un certain temps, et oui, cela a causé des problèmes) ; il avait aussi décidé de la façon dont les droits d'accès devaient être accordés, en fonction de la confiance qu'il portait personnellement au candidat. La « confiance », dans ce cas, se fondait sur une seule chose : non pas le nombre de membres de la communauté qui se portait garant pour cette personne, ni depuis combien de temps cette personne était un contributeur actif et de confiance pour le projet, ni même depuis combien de temps il connaissait lui-même cette personne dans le cadre de ce projet. Au lieu de cela, elle reposait sur la façon dont il connaissait personnellement quelqu'un, ce par quoi il entendait la façon dont il connaissait cet individu en personne. Vous imaginez bien à quel point cela est adapté à une équipe d'administrateurs systèmes disséminée sur toute la planète…

Bien sûr, cet exemple ne fait qu'illustrer la grande difficulté pour un administrateur systèmes *open source* de trouver le juste milieu entre sécurité et capacité. Les grandes entreprises peuvent se permettre d'avoir du personnel redondant, et ce, même si le travail se répartit selon différentes responsabilités ou domaines de sécurité. La redondance est importante. Mais qu'en est-il si la seule possibilité d'avoir une redondance pour l'administrateur systèmes est de prendre la première personne se présentant au hasard sur votre canal IRC ou une personne quelconque proposant son aide ? Comment pouvez-vous raisonnablement avoir confiance en cette personne, ses capacités et sa motivation ? Malheureusement, seuls les contributeurs principaux du projet ou une petite partie d'entre eux peuvent savoir quand la bonne personne se présente en utilisant le même modèle de toile de confiance[3] qui sous-tend une grande partie du reste du monde *open source*. L'univers des projets *open source*, leurs besoins et les personnes qui veulent contribuer à un projet particulier forment une extraordinaire diversité ; par conséquent, la dynamique humaine, la confiance, l'intuition et la manière d'appliquer ces concepts à un projet open source sont de vastes sujets, bien au-delà de la thématique de ce court article.

Une chose importante a cependant facilité la découverte de cette ligne d'équilibre sécurité/capacité : l'essor des systèmes de gestion de versions distribués, ou DVCS (NdT : Distributed Version Control System, système de gestion de version distribué). Auparavant, les contrôles d'accès étaient primordiaux car le cœur de tout projet *open source* — son code source — était centralisé. Je me rends bien compte que beaucoup doivent penser : « Jeff, tu devrais pourtant le savoir, le cœur d'un projet, c'est sa communauté, pas son code ! ». Ma réponse est simple : les membres de la communauté vont et viennent, mais, si quelqu'un fait accidentellement un « rm -rf » sur tout l'arbre du système de gestion de versions de votre projet et que vous manquez de sauvegardes, combien de ces membres de la communauté vont continuer à s'investir dans le projet et aider à tout recommencer à zéro ? (Mes propos se basent sur une histoire vraie dans laquelle un membre de la communauté saoul qui s'énervait à déboguer un bout de code, lança un « rm -rf » sur toute sa contribution, avec l'intention de supprimer tout le code du projet. Par chance, il n'était pas administrateur systèmes et n'avait donc pas accès au dépôt central, et il était trop saoul pour se rappeler qu'il travaillait seulement sur une copie du projet.)

Le code du projet est son cœur ; les membres de sa communauté en sont l'énergie vitale. Privé de l'un ou de l'autre, vous aurez du mal à garder un projet vivant. Avec un logiciel de gestion de version (NdT : VCS pour version control system) centralisé, si vous n'avez pas eu la présence d'esprit de mettre en place un système de sauvegarde régulier, vous pourriez, avec de la chance, ré-assembler l'arborescence complète du code source à partir des différents éléments contribués qu'auront gardés les autres personnes. Mais pour la majorité des projets, l'historique du code est aussi important que le code lui-même et cela, vous l'aurez tout de même entièrement perdu.

Ce n'est plus le cas désormais. Quand tous les clones locaux ont tout l'historique du projet et que des sauvegardes de secours peuvent être effectuées chaque nuit, en lançant une tâche planifiée aussi simple que « git pull », les dépôts centralisés ne sont plus que des outils de coordination. Cela en diminue l'importance de quelques degrés. Le projet doit toujours être protégé contre les menaces aussi bien internes qu'externes : les systèmes non corrigés sont toujours vulnérables à des exploits bien connus. Un administrateur systèmes malveillant peut tout mettre sans dessus dessous, un système d'authentification déficient peut permettre l'entrée de codes malveillants dans la base, et un « rm -rf » accidentel

sur le dépôt central peut toujours coûter cher en temps de développement. Mais ces défis peuvent être surmontés, et à l'ère des serveurs privés virtuels (VPS) abordables et des centres d'hébergement de données, les absences des administrateurs systèmes peuvent également être compensées. (Il vaut mieux cependant s'assurer d'avoir un accès redondant au DNS ! Oh et mettez aussi vos sites internet sur un dépôt vérifié et certifié [DVCS] , et faites des branches pour les modifications locales. Vous me remercierez plus tard.)

Les DVCS permettent la redondance du cœur de votre projet pour trois fois rien, ce qui est une bonne façon d'aider les administrateurs systèmes à dormir la nuit et nous donne l'impression d'être un peu des maîtres du temps. Cela veut aussi dire que si vous n'êtes pas sur un DVCS, arrêtez de lire immédiatement et passez sur l'un d'eux. Ce n'est pas qu'une question d'espace de travail et d'outils. Si vous vous souciez de la sécurité de votre code et de votre projet, vous migrerez.

La redondance du code source est une nécessité, et en général plus vous avez de redondances, plus vos systèmes sont robustes. Il semble aussi évident que vous voulez une redondance de vos administrateurs systèmes ; ce qui vous semblera peut-être moins évident, c'est que l'importance de la redondance ne se joue pas tant en termes de personnes qu'en termes de niveau des compétences. John, l'administrateur systèmes, a travaillé dans des centre de stockage des données et au sein d'entreprises qui avaient des systèmes d'administration redondants mais des niveaux de compétences rigides, définis. Cela fonctionne dans de grandes entreprises qui peuvent payer pour embaucher de nouveaux administrateurs systèmes avec des compétences à la carte. Mais la plupart des projets *open source* n'ont pas ce luxe. Vous devez faire avec ce que vous avez. Cela veut dire bien sûr que, pour que l'administration systèmes soit redondante, une solution — et c'est parfois la seule — consiste à répartir la charge : d'autres membres du projet prenne chacun une ou deux compétences jusqu'à ce qu'il y ait redondance.

Il n'y a guère de différence entre le côté développement et le côté créatif d'un projet ; si la moitié de votre programme est écrite en C++, l'autre moitié en Python, et qu'un seul développeur sait programmer en Python, son départ du projet provoquera de gros problèmes à court terme et pourrait aussi causer de sérieux problèmes à plus long terme. Encourager les développeurs à se diversifier et à se familiariser avec d'autres langages, paradigmes, bibliothèques, etc. entraîne que chacun de vos développeurs gagne en valeur ; cela ne devrait pas choquer : l'acquisition de nouvelles compétences est le résultat d'un

apprentissage qui se poursuit tout au long de la vie, et un personnel mieux formé a aussi plus de valeur. (Cela rend aussi le curriculum vitae de chacun plus attractif, ce qui devrait être une bonne motivation.)

La plupart des développeurs *open source* que je connais considèrent comme un défi et un plaisir de s'aventurer sur de nouveaux territoires : c'est justement ce genre d'état d'esprit qui les a menés à développer de l'*open source* au départ. De même, les administrateurs de systèmes *open source* sont une denrée rare, et ne peuvent se permettre de s'enliser dans une routine. De nouvelles technologies intéressantes pour les administrateurs systèmes apparaissent constamment et il existe souvent de nouvelles façons d'utiliser des technologies actuelles ou anciennes afin de renforcer l'infrastructure ou d'améliorer leur efficacité.

John n'était pas un bon candidat parce qu'il apportait peu de valeur ajoutée ; et il apportait peu de valeur car il n'était jamais allé au-delà des limites du rôle qui lui était attribué. Les administrateurs systèmes *open source* qui tombent dans ce piège ne nuisent pas seulement au projet dans lequel ils sont impliqués sur le moment, ils réduisent leur valeur pour d'autres projets utilisant des technologies d'infrastructure différentes et qui auraient vraiment besoin d'un coup de main ; cela diminue les capacités globales de la communauté *open source*. Pour un administrateur de logiciel libre efficace, il n'existe pas de zone de confort.

1. http://www.kde.org/ ^
2. http://www.tomahawk-player.org/ ^
3. http://fr.wikipedia.org/wiki/Toile_de_confiance ^