Les entrailles de my

Predrag.Viceic@epfl.ch, EPFL - Domaine IT, concepteur de my.epfl

The article unveils the brewing secrets of the new release of my.epfl service. Cheers!

Voici enfin le voile levé sur l'architecture du service my.epfl. Âmes sensibles s'abstenir !

Sept ans déjà que le projet **my.epfl** fournit aux utilisateurs un outil maison de partage de fichiers et d'agendas. La nouvelle version, le fruit de dix-huit mois de développement, a été mise en production au début de l'été. La température retombée, il est venu le temps de dévoiler quelques secrets de brassage de cette nouvelle mouture.

Les trois fonctions principales de my.epfl sont le partage de documents, le partage d'agenda et la gestion des groupes. Ces trois fonctions sont accessibles principalement depuis l'interface Web, qu'on trouve à l'adresse **my.epfl.ch**.

Le portail Web

Le portail Web sert de conteneur aux trois applications qui sont **Documents**, **Agendas** et **Groupes**. Il fournit l'entête du site, s'occupe de la gestion des onglets, de l'authentification et détermine, selon les critères basés sur les groupes, la disponibilité des onglets pour les personnes authentifiées.

Le portail, basé sur uPortal v3.2.4, est un produit *open source*, fourni sous licence Apache. Il implémente les normes Portlet v1 (JSR-168) et Portlet v2 (JSR-286). Ces normes permettent l'intégration d'applications Web *(servlets)* dans un environnement de portail. Ainsi, une application Web respectant la norme peut être intégrée sans modification dans n'importe lequel des environnements conteneur JSR-168/286. Le placement sur la page, l'état de l'application Web, ainsi que les identifiants des objets de l'arbre DOM générés par l'application sont gérés par le portail. L'implémentation du standard est fournie par Pluto 1.0 / 2.0, le conteneur de *portlet* du projet Apache qui est également l'implémentation de référence de la norme en question.

L'avantage de uPortal, en plus de sa généralisation en tant que portail universitaire ayant une grande communauté d'utilisateurs et de développeurs, est son extrême souplesse lors de l'intégration dans les environnements IT existants. Ainsi, lors d'implémentation de la charte graphique de l'École, il n'a pas été nécessaire d'écrire une ligne de code. Tout s'est fait en manipulant les CSS et les fichiers XSL décrivant la structure visuelle du site.

Lors de l'intégration de l'authentification Téquila, le SSO *(single sign-on)* de l'École, il a suffi de déployer le filtre Téquila légèrement modifié pour l'occasion. Toute la

../.. Suite page 11

../.. Suite de la première page

structure de l'authentification existante dans uPortal et basée sur CAS *(Central Authentication System*, l'autre grand système de SSO) a pu alors être remplacée par Téquila, sans modifier le code de uPortal.
La gestion des autorisations, basée sur les personnes et les groupes a pu sans autre être intégrée avec notre infrastructure LDAP.

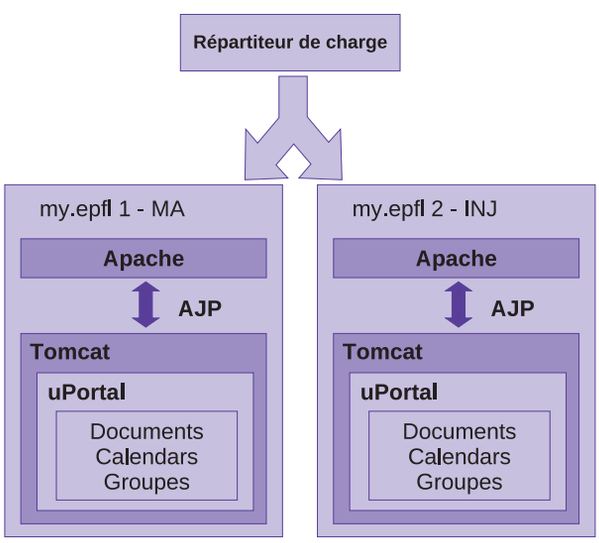
Serveur d'applications: Tomcat

L'application uPortal s'exécute sur deux serveurs physiques, le premier dans le bâtiment MA et l'autre dans le nouveau data center de l'EPFL dans le bâtiment INJ. Les serveurs exécutant la distribution RedHat Linux font tourner le conteneur d'applications Web Tomcat 6, un autre grand projet de la fondation Apache. Les disques réseau du projet Sanas de l'EPFL servent comme stockage des logs système et donnent un moyen de communication entre les serveurs. Ces montages sont accédés en protocole NFS v3.

Serveur Web: Apache

Sur le chemin de l'utilisateur au serveur, le service Tomcat est précédé par le service Apache, qui redirige toutes les requêtes entrantes en utilisant le protocole AJP. AJP *(Apache Jserv Protocol)* est un protocole permettant la redirection des requêtes Web entre le serveur Web (Apache) et le serveur d'applications (Tomcat). Il va sans dire que les deux services sont exécutés sur la même machine physique. Apache en *front-end* peut paraître superflu et complexifiant inutilement l'architecture. La plupart du temps cette affirmation est vraie. Toutefois, en cas d'attaque du type déni (distribué) de service, il est très facile de bloquer les clients fautifs au niveau du service Apache, tâche beaucoup moins évidente à faire au niveau du serveur d'applications.

Le répartiteur de charge

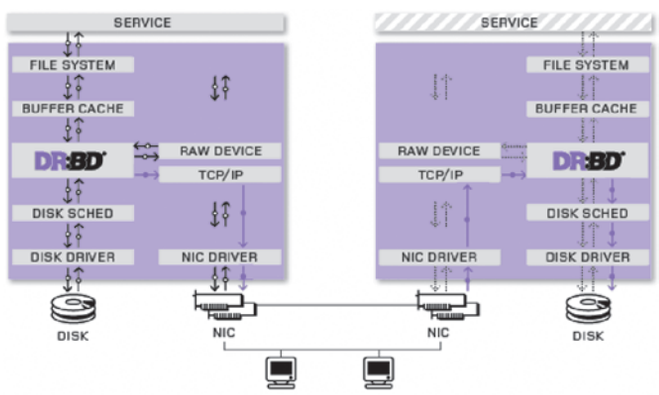


Architecture de my.epfl

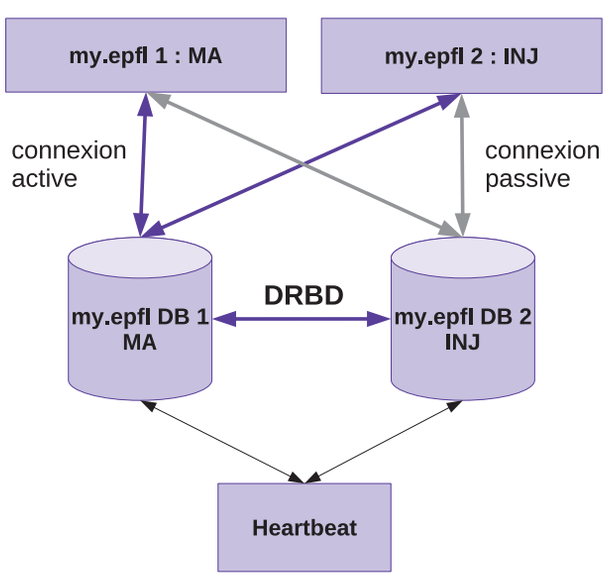
La dernière pièce d'infrastructure entre les serveurs d'application et l'utilisateur final est le répartiteur de charge de l'École, le Cisco ACE *(Application Control Engine)*. Appelé par son petit nom: le *content-switch*, le Cisco ACE exécute deux tâches pour le projet my.epfl. Tout d'abord il repartit les requêtes *(round-robbin)* entre les deux serveurs se trouvant dans les deux salles. Finalement, il s'occupe du cryptage/décryptage de la communication lorsque celle-ci est établie à travers le protocole SSL. Ceci est toujours le cas avec le projet my.epfl. Anecdotiquement, notons que le premier *content-switch* de l'École a été acquis pour les besoins du projet my.epfl, le hardware servant à beaucoup d'autres services depuis lors. Comme nous le verrons plus tard, le répartiteur de charge effectue une surveillance des serveurs et décide, selon des critères bien définis, quand un serveur ne peut plus répondre aux requêtes. Dans ce cas il arrête de rediriger le trafic vers ce serveur, en attendant que le problème soit résolu.

La base de données

La base de données est de couleur PostgreSQL v8, fournie en licence PostgreSQL, semblable aux licences BSD ou MIT. Les bases de données, une active et une passive sont, à l'image des serveurs d'applications, reparties dans les deux salles. La réplication des données se fait à l'aide du protocole DRBD *((Distributed Replicated Block Device – périphérique en mode bloc répliqué et distribué)*, une implémentation *open source* (GPL v2) du *mirroring* par bloc à travers le réseau. DRBD pourrait être comparé à du RAID-1 réseau.



Architecture DRBD



Architecture base de données

Cette infrastructure nous permet de nous assurer que le système continuera de fonctionner en cas de perte d'un des deux serveurs. C'est le logiciel *open source* Heartbeat qui détecte l'indisponibilité

du serveur actif et qui lance la modification de la configuration réseau ainsi que la configuration DRBD, redirigeant ainsi les requêtes vers la base de données fonctionnelle. Comme les deux partitions contenant les données sont synchrones grâce au DRBD, le fonctionnement peut reprendre sans perte d'informations.
Le choix de PostgreSQL vs. MySQL s'est imposé à cause d'une implémentation beaucoup plus mature des requêtes transactionnelles, un luxe peu usité dans les projets Web, mais indispensable pour la partie documents de my.epfl.

LDAP

Le système d'autorisation sur lequel reposent les applications du service my.epfl est le serveur LDAP de l'École. L'EPFL possède deux services LDAP, l'un du type annuaire, reproduisant la structure hiérarchique des unités administratives, et l'autre, SCOLdap (Ldap des Services Collaboratifs) présentant une structure à plat, beaucoup plus facilement intégrable avec les différents outils du marché. SCOLdap a été conçu il y a sept ans pour les besoins du projet my.epfl, mais depuis il est utilisé par de nombreux autres services de l'École. Le service LDAP est actuellement déployé sur quatre serveurs positionnés dans deux salles distinctes et précédé par le *content-switch*. Ainsi nous pouvons garantir une haute disponibilité et une résilience à la charge de cet ingrédient essentiel pour beaucoup d'autres applications de l'École.

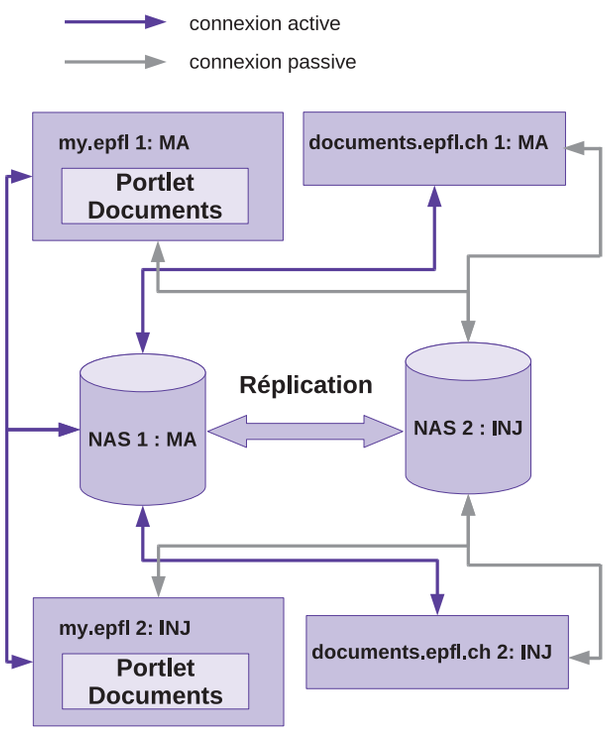
Documents

Portlet Documents

L'application **Documents**, accessible depuis l'onglet du même nom sert à manipuler l'espace de stockage collaboratif de my.epfl. L'application utilise l'API Xythos v7, la seule partie de my.epfl à être en licence propriétaire. Le produit Xythos, utilisé dans les produits SAP, Oracle,... fournit la couche métier de l'onglet **Documents**. C'est cette API qui communique avec la base de données **Documents**, hébergée sur les deux serveurs PosgreSQL mentionnés plus haut.
C'est également l'API Xythos qui manipule les fichiers stockés sur le NAS et accédés avec le protocole NFS 3. Cette API s'occupe des versions, du verrouillage des fichiers et de beaucoup d'autres fonctions. Xythos communique également avec Lucene, un projet Apache sous licence Apache, afin d'indexer les documents pour que ceux-ci soient retrouvables par la recherche full-text de my.epfl.
La partie interface de l'onglet **Documents** a été développée ici, au KIS, en utilisant le framework ZK. ZK est le produit de la firme taiwanaise Potix. ZK existe en double licence, LGPL et Licence commerciale. ZK se charge de fournir l'interface Ajax des onglets **Documents**, **Groups** et **Agenda** du projet my.epfl. En plus de fournir un environnement de développement moderne, agréable et très efficace, ZK offre une intégration de la norme JSR-168 (Portlet). Ceci nous permet de développer les applications intégrables dans diverses solutions de portail.
Ni la norme Portlet v1, ni la norme Portlet v2 n'ont pas été conçues pour les requêtes du type Ajax. C'est pour cela que le framework ZK fournit un filtre Tomcat qui intercepte tous les appels asynchrones et les redirige vers la *portlet*.

Les subtilités du développement des interfaces hautement dynamique telles que celles de my.epfl dépassent largement le cadre de cet article. Je note toutefois que nous utilisons Spring pour l'injection de dépendances (Inversion de contrôle), Hibernate pour la couche persistance, C3P0 comme *pooler* des connexions BDD, Ehcache comme implémentation des caches, pour un total de plus d'une centaine de librairies externes.
La partie documents développée à l'EPFL, qui exclut l'API Xythos et toutes les librairies externes, correspond à environ 16'000 lignes de code java et plus de 1'600 lignes de markup ZK. Ce qui, entre nous, n'est pas beaucoup.

Le serveur WebDAV



Documents. Les connexions à la base de données ne sont pas représentées pour plus de clarté.

En plus de fournir l'API utilisée pour développer l'interface Web, Xythos fournit également une excellente implémentation du protocole WebDAV: le service documents.epfl.ch. Ce service est déployé sur deux serveurs physiques, hébergés dans les deux data-centers. Notre ami le répartiteur de charge sert de front-end à l'application. Le serveur WebDAV fournit l'accès aux fichiers sur l'interface Web documents.epfl.ch. Il permet également le montage réseau de l'espace de documents sur les machines des utilisateurs. Ainsi nous pouvons manipuler les documents non seulement à travers l'interface my.epfl.ch, mais également en utilisant le gestionnaire de fichiers natifs d'une machine Linux, OSX, Windows, mais également Android et IOS.
Le serveur WebDAV est connecté à la même base de données que tout le reste, et accède aux mêmes montages NAS que l'API de l'interface my.epfl. Le protocole AJP étant peu conciliant avec le protocole WebDAV, nous ne déployons pas de serveurs Apache devant les serveurs Tomcat dans le cas du service documents.epfl.ch.
En utilisant le protocole WebDAV, d'autres applications telles que wiki.epfl.ch, blogs.epfl.ch ou encore inform.epfl.ch déposent les fichiers sur l'infrastructure des documents de my.epfl. Les applica-

tions wiki et blogs vont jusqu'à modifier les droits d'accès sur les fichiers au gré des modifications des droits d'accès sur les wikis ou les blogs correspondants.

Agendas

Bedework

L'application de gestion des calendriers est accessible depuis l'onglet **Agenda** du portail my.epfl. Afin de fournir les fonctionnalités nécessaires, nous utilisons l'API du projet Bedework, v3.4, fournie en licence BSD modifiée.

Bedework est un projet de l'Université de Washington. En 2003, le projet est repris par Rensselaer Polytechnic Institute. Actuellement, le service Bedework est déployé dans une douzaine d'universités à travers le monde, dont Yale, Duke, Université de Navarre et même la bourse allemande *(market data)*.

Bedework fournit une API puissante, mais malheureusement l'interface Web laisse à désirer. Ce petit détail ne nous a pas découragés, c'est donc ici, au KIS, que la première mouture de l'application **Agenda** a été développée en utilisant l'API Bedework et le framework JSF. JSF, pour *Java Server Faces*, est un cadriciel (sic) de la fondation Apache fournissant les outils pour le développement d'interfaces Web.

Une fois la première version de l'application **Agenda** achevée et mise en production à l'EPFL, le projet fut repris par le Centre Informatique de l'Université de Lausanne et déployé en tant que système d'agendas collaboratifs de l'Université.

C'est là-bas, chez nos collègues de l'UNIL, que la nouvelle version a vu le jour sous le nom de MyAgenda. Cette version continue à utiliser l'API Bedework, mais l'interface a été conçue en utilisant le framework ZK, beaucoup plus moderne que le JSF.

Le développement de l'UNIL a été ensuite repris par l'EPFL, où nous avons implémenté notre charte graphique et refait l'ergonomie de l'application. Depuis, ces améliorations ont été introduites dans la version déployée à l'UNIL. À l'Université de Lausanne, MyAgenda est déployé dans le portail académique my.unil, qui utilise le conteneur de *portlet* Jahia. La version actuelle déployée dans my.epfl est donc le fruit d'une collaboration entre les deux institutions s'étendant sur plusieurs années.

L'API Bedework communique avec la base de données (repartie, DRBD, oui c'est toujours la même…) et fournit ainsi toutes les fonctionnalités nécessaires à un système d'agendas partagés. La manipulation des évènements, des calendriers, des vues et la gestion des droits d'accès se font par l'API Bedework, que nous avons modifiée pour supporter notre infrastructure d'autorisations (SCOLdap).

Notre collaboration avec le Rensselaer Polytechnic Institute nous permet de supprimer à chaque nouvelle version une partie de nos ajouts ou modifications, celles-ci se retrouvant dans la version originelle de l'API Bedework. Des rumeurs racontent que le Centre Informatique de l'Université de Lausanne se prépare à porter l'application à la version Bedework 3.9, ce qui permettra, entre autres, une configuration beaucoup plus aisée des appareils mobiles.

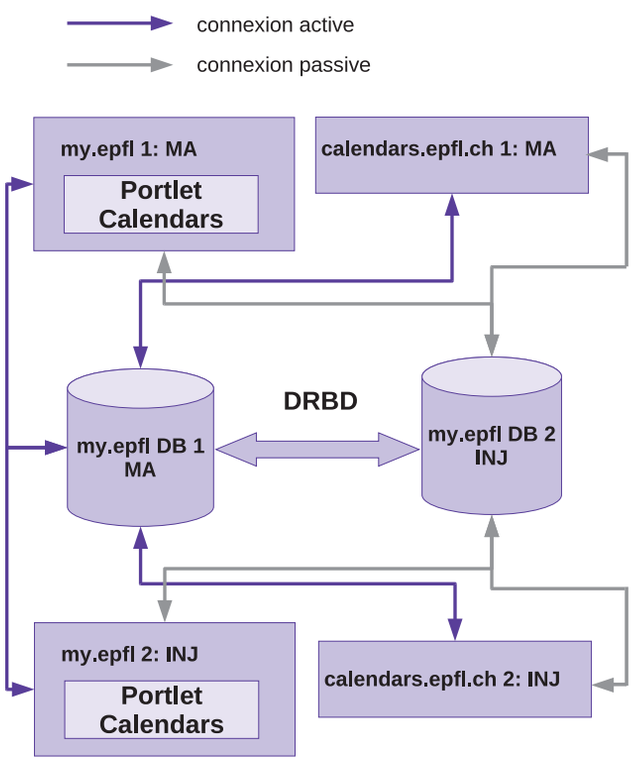
Le serveur CalDAV

Le projet Bedework, en plus de fournir une API très puissante, fournit également un serveur implémentant la norme CalDAV. Une version légèrement modifiée de celui-ci est déployée en tant que service calendars.epfl.ch.

Le service **Calendars** permet la manipulation des évènements des agendas du projet my.epfl avec les clients desktop ou mobiles. Tout périphérique ou application d'agenda supportant la norme CalDAV, peut donc y accéder, permettant ainsi l'intégration des agendas de my.epfl dans les applications habituelles des utilisateurs. Il va également sans dire que la seule application notable ne supportant pas le protocole CalDAV est Microsoft Outlook.

Le service calendars.epfl.ch est déployé sur deux serveurs physiques installés dans les deux salles en MA et en INJ. Le répartiteur de charge se charge de la distribution balancée entre les deux instances.

La partie **Agenda** développée à l'UNIL et EPFL, ce qui exclut l'API Bedework et toutes les librairies externes, correspond à environ 21'000 lignes de code java et environ 1'300 lignes de markup ZK.

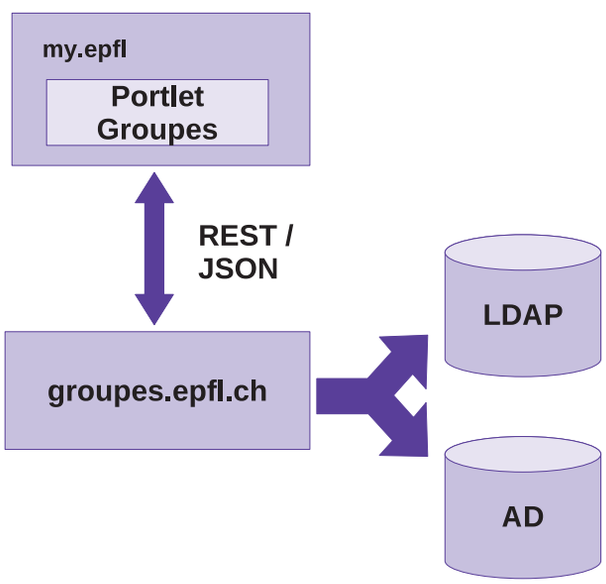


Calendars

Groupes

La dernière application du projet my.epfl est l'application de gestion des Groupes. Cette application permet la création et la manipulation des groupes d'utilisateurs. Comme de coutume, il s'agit là de nouveau d'un service à l'architecture n-tiers. Le service de groupes, accessible sur l'interface groups.epfl.ch a une vie indépendante du projet my.epfl. Développé également au KIS, ce service manie la base de données des groupes et leurs membres, et se charge de la réplication de cette structure vers les services tels que LDAP ou Active Directory.

My.epfl utilise intensivement les groupes comme source pour les autorisations, il est donc normal qu'il fournisse une interface pour leur manipulation. L'interface est développée avec le framework ZK et utilise les services Web pour communiquer avec le service groupes.epfl.ch.
Ainsi, chaque manipulation de groupe via l'interface my.epfl.ch génère les requêtes vers le service groupes.epfl.ch. Ces requêtes respectent (approximativement) la norme REST et encodent les informations à l'aide de la syntaxe JSON.
L'ancienne version de my.epfl utilisait XML-RPC pour communiquer avec le service des groupes. Nous avons décidé, et le temps nous donne raison d'abandonner cette norme bien après que les autres l'aient fait.
La partie **Groupes** correspond à environ 12'000 lignes de code java et environ 1'700 lignes de markup ZK.



Groupes

Services Web

La communication entre my.epfl et le service des groupes ne s'effectue pas dans un seul sens. En effet, dès qu'un groupe est créé, et ceci indépendamment du fait que l'action soit faite à travers my.epfl.ch ou groupes.epfl.ch, les services Web déployés sur les serveurs de my.epfl sont notifiés par l'application groupes.epfl.ch.
Cette voie de retour permet de créer, supprimer ou renommer les espaces de stockage des groupes de my.epfl, reflétant ainsi les modifications faites au niveau de l'application de gestion des groupes.
En plus des services Web permettant la manipulation des structures de stockage, le service my.epfl déploie également d'autres services servant à l'*upload* depuis une page Web, affichage de la structure des dossiers / agendas dans une page Web et affichage des disponibilités pour en citer que les plus utilisés. De nouveaux services sont ajoutés régulièrement, permettant ainsi une meilleure intégration du projet my.epfl dans la structure IT de l'École.

Déploiement & Monitoring

Puppet et le déploiement

Le projet my.epfl est déployé sur huit serveurs, sans compter les serveurs de développement et de *staging*. Il serait impossible de gérer une telle architecture d'une manière efficace sans mettre en œuvre les outils spécialisés. Ainsi, le déploiement des serveurs (OS, couche service) se fait à l'aide du logiciel *open source* Puppet, fourni en licence Apache 2.0.
Notre partenaire dans l'aventure, l'entreprise Camp to Camp présente sur le campus de l'EPFL, cuisine nos recettes Puppet, qui sont une sorte de fichiers de déploiement. Les clients Puppet sur chaque serveur interprètent et appliquent ces recettes sur la configuration logicielle. Ainsi il devient un jeu d'enfant de déployer un nouveau serveur, soit pour en faire un serveur de test, pour pallier à une panne hardware, ou pour soutenir une éventuelle augmentation de charge.
Avant, ce genre d'opérations étaient faites à la main et pouvaient prendre plusieurs heures, voire des jours. Actuellement, ce n'est plus qu'une question de quelques dizaines de minutes.
En plus de nous assister lors de la phase d'installation et déploiement, Puppet permet les modifications centralisées de la configuration des serveurs déjà en service. Ceci s'avère être très pratique lorsqu'on doit modifier un paramètre sur tous les serveurs, ou ajouter un binaire particulier sur un service.

Nagios et le monitoring

Dans tout service potentiellement accessible en mode 24/7/365, il est primordial de pouvoir détecter les pannes avant les utilisateurs. La surveillance automatisée *(monitoring)* des serveurs devient alors indispensable. Le projet my.epfl utilise le logiciel Nagios, en licence GPL, pour assumer cette tâche.
Nagios, configuré et bichonné par Camp to Camp, effectue en continu toute une série de vérifications sur l'ensemble des serveurs my.epfl. Il vérifie l'occupation du disque local et celui du montage NAS. Nagios vérifie la disponibilité de la base de données, ainsi que le nombre de connexions clientes sur celle-ci. Il vérifie l'existence du processus httpd (Apache) ainsi que celle du processus Tomcat.
En remontant l'infrastructure, Nagios vérifie la disponibilité du service Webdav en tentant en continu l'accès à un fichier donné. Il vérifie également si les services sont lancés avec les bons droits et toute une suite d'autres paramètres permettant de détecter un problème au plus tôt. Dès que le problème est détecté, les personnes en charge de l'infrastructure sont notifiées, selon leur rôle dans la résolution du problème en question.

Répartiteur de charge et le monitoring

Nagios n'est pas le seul à surveiller les services my.epfl. Le répartiteur de charge effectue également une série des tests sur les services my, documents et calendars. Il teste la réponse du serveur au niveau de la connexion TCP, mais pas uniquement. Le ACE appelle régulièrement un service Web, conçu pour l'occasion et qui vérifié l'empreinte mémoire de la machine virtuelle Java, ainsi que la quantité de *threads* concurrents s'exécutant sur la machine. Les valeurs hors limite signalent un problème d'emballement de la machine virtuelle ou une charge trop grande.

Une fois l'emballement détecté, le service Web modifie sa réponse au *content-switch*, ce qui provoque la sortie du serveur de la ferme. Les requêtes ne sont plus envoyées vers ce serveur, mais uniquement vers son *alter ego* fonctionnel.
Le répartiteur de charge continue néanmoins à appeler le service Web du serveur dans les choux, en espérant une réponse positive. Une fois les choses tassées, la récupération de mémoire ayant été faite par le ramasse-miettes de la JVM, le service Web notifie le répartiteur du retour à la normale. Le serveur guéri est alors remis dans le cluster, avec comme seule trace de la panne, quelques mails matinaux automatiques nous notifiant du dysfonctionnement. Le service Web en charge de la surveillance de la machine virtuelle Java utilise le protocole JMX *(Java Management eXtensions)*.

Backups

Il y en a beaucoup. Partout.

Tâches de maintenance

Plusieurs tâches périodiques s'assurent du bon fonctionnement des services. Elles vérifient l'attribution correcte des droits d'accès, configurent les quotas et vérifient la présence des espaces des groupes et des agendas de groupes. Il y en a pour tous les goûts. Certaines vérifient l'état de la machine virtuelle et notent les statistiques afin de permettre l'analyse des patrons d'utilisation d'un service particulier. Elles évaluent en temps réel l'usage du système de stockage du service **Documents** et construisent les statistiques de son utilisation. Elles permettent également de corriger les couacs des opérations effectuées à la volée. En bref c'est en tout une dizaine de scripts qui sont exécutés en permanence et qui s'assurent de la cohérence des données et de la bonne marche du service.

Créature de Frankenstein

Avant que d'autres le fassent, je ferai la comparaison moi-même: le service my.epfl utilise une multitude d'applications diverses, de librairies logicielles, de services et de frameworks, cousus ensemble dans un but commun. Parfois, ce service me fait penser à la créature de Frankenstein, à quelques différences près: une apparence plus jolie, une durée de vie plus longue, consommant beaucoup plus d'électricité et je l'espère beaucoup plus utile. ■