Déploiement logiciel open source et multiplateforme
Acte2

Guillaume Ferry
Direction du Numérique – Sous-direction Services aux Usagers
Campus Aiguillettes
54500 Vandoeuvre-les-Nancy

Nicolas Rogier
Direction du Numérique – Sous-direction Services aux Usagers
Cellule Exploitation & Projets
24-30 rue Lionnois
54000 Nancy

Résumé

La solution de déploiement de logiciels que nous vous avons présentée lors des JRES 2019 a retenu votre attention ? Alors ne manquez pas la saison 2 !

Largement adoptée par les équipes de proximité de l'Université de Lorraine et préconisée par la direction du numérique, notre solution pilote désormais le cycle de vie de plus de 11000 machines réparties sur 19 sites géographiques.

Venez découvrir lors de nos sessions « Posters » comment nous avons répondu aux nouvelles problématiques soulevées par cette montée en charge :

- *Évolution de l'infrastructure serveurs pour garantir performance et haute disponibilité du service*
- *Automatisation du développement et de la mise en production des paquets*
- *Uniformisation des configurations et des pratiques à l'échelle de l'établissement*

En parallèle, une session de démonstration vous permettra de tester les nouvelles fonctionnalités de ce service.

Suite aux JRES 2019, vous avez été nombreux à exprimer votre intérêt. L'ouverture de ce service pour la communauté de l'enseignement supérieur et de la recherche est l'une de nos nouvelles perspectives. Et si vous deveniez acteur de la saison 3 ?

Mots-clefs

Puppet, Nexus, Gitlab, Chocolatey, APT, Homebrew, Rundeck, gestion de parc, Devops, déploiement logiciel, Keepalived, Haproxy, Infrastructure as Code

1 Contexte

La solution de déploiement logiciel présentée lors des JRES 2019 [1] est devenue un outil incontournable au sein de la sous-direction des services aux usagers de l'Université de Lorraine. Plus de 11000 machines réparties sur 19 sites géographiques sont désormais administrées avec cette solution, préconisée par la direction du numérique.

La montée en charge du service a soulevé de nouvelles problématiques et de nouveaux besoins. Nous vous proposons d'illustrer les évolutions apportées à notre infrastructure à l'aide d'un poster et de mettre en avant la simplicité et l'efficacité du service au travers d'une démonstration.

2 Problématiques

Au cours de ces dernières années, la démocratisation du télétravail a fortement influencé la politique d'achat de matériel informatique au sein de l'Université de Lorraine. L'ordinateur portable a brusquement remplacé le poste fixe ; améliorant la mobilité, mais réduisant en contrepartie les fenêtres d'interventions pour les équipes de proximité.

Pour faire face au pic d'activité généré par les nombreuses commandes de matériel mobile et pour répondre aux problématiques posées par ces nouveaux usages, les équipes de sites ont largement adopté la solution.

Montée en charge et pérennité du service

L'augmentation constante du nombre de postes clients montrait les limites de l'infrastructure, notamment sur sa disponibilité, ses performances et son exploitation.

2.1 Configuration centralisée

En nous appuyant sur nos outils de supervision (Zabbix, Netdata, *PuppetDB performance dashboard*), nous avons constaté que les ressources allouées au service *Puppet* ne seraient bientôt plus suffisantes pour assurer la *compilation des catalogues* des postes clients. Les préconisations de *Puppetlabs [2]* quant au dimensionnement du service, mettaient en évidence le besoin de renforcer notre infrastructure *Puppet,* où les rôles *Puppetserver* et *PuppetDB* reposaient sur une seule VM.

2.2 Dépôts logiciels

L'augmentation constante du nombre de postes clients durant ces 4 dernières années a également impacté les performances de nos 2 dépôts logiciels *Nexus,* jusqu'alors configurés en mode *actif/passif*. Nos outils de supervision et l'analyse des logs nous ont permis de mesurer et d'identifier les raisons des ralentissements constatés en production. Durant les périodes les plus chargées de la journée, le flux des requêtes sur *l'API Nexus* dépasse régulièrement les 100 requêtes par seconde. La charge du serveur frontal (CPU, bande passante) atteignait régulièrement son niveau maximal lors de la mise à disposition de nouvelles versions de paquets. Nous avons donc décidé de faire évoluer le schéma de ce service qui atteignait ses limites.

Développement et intégration continue

L'automatisation des installations et des mises à jour logicielles offre un gain de temps considérable pour les gestionnaires de parc informatique. En contrepartie, un investissement non négligeable est nécessaire pour maintenir et étoffer le catalogue logiciel de l'université, qui ne cesse de s'accroître. En mars 2022 ce sont plus de 800 paquets *Chocolatey*, *Debian*, et *Homebrew*, qui sont mis à disposition dans les dépôts de la solution pour les plateformes *Windows*, *Linux* et *macOS*. Nous avons dû faire face à une forte demande de la part des utilisateurs de la solution pour créer et mettre à jour les applications déployées sur leur parc. Des actions sur le plan technique mais également organisationnel, ont été nécessaires pour maintenir la dynamique qui a contribué au succès de la solution.

Émergence de nouveaux besoins

L'harmonisation des pratiques liées au déploiement de logiciels a ouvert de nouvelles perspectives, faisant naître le besoin d'appliquer une politique de gestion commune sur de nouveaux périmètres transverses (directions, laboratoires, etc..). En concertation avec la direction du numérique et les responsables d'équipes de site, nous avons étudié la possibilité de faire évoluer l'interface d'administration pour matérialiser des périmètres fonctionnels et en déléguer la gestion.

3 Solutions

Optimisation de l'infrastructure

3.1 Configuration centralisée

Pour traiter les problématiques d'équilibrage de charge et de haute disponibilité du service *Puppet*, nous avons créé une nouvelle infrastructure parallèlement à l'existante, en tenant compte notamment des recommandations de *Puppetlabs*. Ce mode opératoire présentait l'avantage de pouvoir travailler sereinement et en toute indépendance de la production. Par la même occasion, nous avons pu procéder aux montées de versions majeures des différentes briques logicielles utilisées, comme *Puppetserver, PuppetDB, postgreSQL, etc...* Le moment venu, la redirection des communications agents/serveurs vers la nouvelle infrastructure serait assurée par un module *Puppet* dédié à la migration.

Cette nouvelle infrastructure *Puppet* est composée de 7 serveurs :

– L'un héberge l'autorité de certification nécessaire à l'authenticité des communications entre agents et serveurs.

– 4 autres, sont dédiés à la compilation des catalogues (*compilers*), et forment un cluster HAProxy de niveau 3. Ce cluster est joignable via une IP virtuelle (*VIP*), implémentée à l'aide de Keepalived*[3]*.
Nous avons choisi d'associer les 2 briques Open Source *HAproxy* et *Keepalived* pour leur simplicité de mise en œuvre et leur efficacité.

– Les 2 derniers hébergent le service *PuppetDB* dans un cluster *PostgreSQL* en mode *actif/passif*.

Le code *Puppet* est maintenu dans un dépôt *git*, répliqué sur les *compilers Puppet*. Le mécanisme pour assurer cette synchronisation est codé dans un *hook git*.

Les modules *Puppet* sont quant à eux synchronisés à l'aide de *R10k*, qui est l'outil préconisé par P*uppetlabs* pour ce type de tâches.

La mise à l'échelle du service *Puppet* est largement documentée sur le site de *Puppetlabs*. Cependant, la configuration des *compilers* (quantité de *compilers* à exécuter, paramètres d'instances *Puppet/JRuby*, *max_active_instance*, *ReservedCodeCache*, *java_heap_size*, *etc*....) dépend de nombreux paramètres tels que les ressources matérielles allouées aux *compilers*, le nombre de clients et leurs intervalles de compilation, etc... Pour tenir compte de l'augmentation de la charge du service, nous avons révisé ces paramètres à plusieurs reprises pour conserver un fonctionnement optimal.

Côté clients, cette nouvelle infrastructure a nécessité la modification des paramètres de nos agents, pour distinguer le service de compilation, du service d'autorité de certification. (*puppet.conf* : Paramètres *server, ca_server*). Nous avons également retardé, de manière contrôlée, le démarrage de l'agent *Puppet* pour lisser la charge des *compilers* (*puppet.conf* : paramètres *splay, splay_limit*).

Développement et intégration continue

L'harmonisation des pratiques liées au déploiement logiciel, notamment avec l'utilisation exclusive des gestionnaires de paquets, induit une augmentation de l'activité autour du développement de paquets. Nos pratiques ne permettaient plus de maintenir raisonnablement un catalogue logiciel tel que celui d'une université. Aussi, nous avons mené plusieurs actions visant à améliorer la qualité du code et accélérer les délais de mise en production.

– Nous avons tout d'abord cherché à automatiser les étapes reproductibles de la phase de développement de paquets. Le passage en préproduction est conditionné par le résultat de nombreux tests réalisés à l'aide de pipelines *GitLab* (analyse syntaxique et antivirale, installation, désinstallation, etc...).

– Un transfert de compétences a permis aux informaticiens de site, qui en ont exprimé le souhait, de devenir autonomes dans le développement de paquets. On compte aujourd'hui une vingtaine de mainteneurs sur le périmètre de l'université.

– L'équipe en charge de la solution est désormais composée de 6 informaticiens pour assurer l'exploitation et faire évoluer le service.

– Pour simplifier les tâches d'administration courantes et les rendre accessibles à n'importe quel membre de l'équipe, nous avons développé un *backend* d'administration basé sur les API de *Nexus*, *Rundeck* et *GitLab*. La majorité des actions réalisées manuellement jusqu'alors, ont été automatisées (passage en production de paquets, création de dépôts *GitLab*, jobs de réplication, d'internalisation, etc...).

Par ces actions, nous avons réussi à diminuer le coût humain de l'exploitation de la solution, tout en fiabilisant la chaîne de développement et d'intégration.

Les tâches d'administration de la solution ont été simplifiées et automatisées, permettant d'intégrer facilement de nouveaux membres au sein de l'équipe.

Le transfert de compétences réalisé auprès des équipes de sites a permis d'impliquer un plus grand nombre de personnes et de diminuer la charge de travail des administrateurs de la solution.

Support de nouveaux cas d'usages

Initialement, la solution a été conçue pour répondre aux problématiques de déploiement logiciel au sein d'un périmètre géographique défini, géré par une équipe de site. Certaines structures, comme les laboratoires ou les directions opérationnelles, peuvent s'étendre sur plusieurs sites géographiques, et sont de ce fait, cogérées par plusieurs équipes.

Nous avons dû adapter l'ergonomie du portail d'administration ainsi que le code *Puppet* pour tenir compte de ces particularités.

Les périmètres sont toujours définis grâce aux *environnements Puppet*, cependant les permissions sur certains environnements peuvent être accordées à plusieurs équipes de sites. Cette évolution nous a également permis de répondre aux réorganisations d'équipes, parfois amenées à intervenir sur plusieurs périmètres géographiques.

D'autre part, nous avons ajouté la possibilité d'utiliser des profils logiciels communs, validés par la direction du numérique et en accord avec les responsables d'équipes de sites.

La mise à disposition de ces profils et la représentation des premiers périmètres opérationnels multi-sites dans la solution, est une étape supplémentaire vers l'harmonisation de la configuration logicielle des postes informatiques de l'université, souhaitée par l'ensemble de la direction du numérique.

4 Bilan

Durant ces 3 dernières années, la solution s'est imposée comme le principal outil de déploiement logiciel auprès des équipes de sites. De nombreuses modifications ont été nécessaires pour adapter la solution à l'échelle de l'établissement et répondre aux problématiques associées.

Une API est en cours de développement pour standardiser et simplifier les communications entre les briques logicielles du service (*Puppet, Nexus, Gitlab, Rundeck*). Celle-ci favorisera le découplage de l'interface (*front-end* / *back-end*) dont les avantages sont nombreux :

- Code source plus facilement maintenable.
- Interopérabilité depuis n'importe quel autre outil capable de réaliser des appels d'API.
- Ouverture vers une architecture micro-services.

Les briques logicielles de la solution sont des applications C*loud-Native* qui s'intègrent naturellement dans un environnement *Kubernetes*. Le portage vers une architecture micro-service est l'opportunité de pouvoir démarrer plusieurs instances du service. L'objectif étant de proposer une offre clé en main à d'autres établissements de l'enseignement supérieur et de la recherche.

Annexes

Contributions

Pour compléter la présentation réalisée lors des JRES 2019, nous vous proposons 2 autres types de contributions pour étoffer notre retour d'expérience :

– Un poster, qui fera le point sur l'état actuel de l'architecture et qui viendra compléter l'article.

– Une ou plusieurs sessions de démonstration pour vous faire découvrir la solution et ses fonctionnalités de manière plus interactive.

Remerciements

Nous tenons à remercier,

– la direction du numérique de l'Université de Lorraine qui nous donne les ressources et le soutien nécessaire pour porter ce projet,

– les équipes de sites qui font vivre la solution en apportant des remarques constructives et en contribuant au développement de paquets,

– le groupe de travail pour sa forte implication dans le projet et plus particulièrement Miguel Coria (Université de Lorraine, Campus Aiguillettes) pour la préparation de la maquette qui sera utilisée pour animer les sessions de démonstration.

Bibliographie

[1] Guillaume Ferry. Déploiement logiciel automatisé et multiplateformes. JRES2019, Dijon, Décembre 2019; https://conf-ng.jres.org/2019/document_revision_4968.html?download

[2] Puppetlabs. Maintaining, tuning and scaling Puppetserver/PuppetDB
https://puppet.com/docs/puppet/6/server/tuning_guide.html
https://puppet.com/docs/puppetdb/6/maintain_and_tune.html
https://puppet.com/docs/puppet/6/server/scaling_puppet_server.html

[3] Alexandre Simon. Haute disponibilité et répartition de charge enfin libérées ! JRES2011, Toulouse, Novembre 2011; https://2011.jres.org/archives/110/index.htm