Les bonnes pratiques DevOps et Application Release Automation

Packaging applicatif, enchaînements et modèle de déploiement standard

ca technologies | Automic™ Livre blanc

TABLE DES MATIÈRES

La complexité : une évolution majeure

Applications et services comprennent aujourd'hui davantage de composants et d'intégrations. S'ajoute à cela une concurrence économique toujours plus acharnée et l'on constate rapidement à quel point la mission des équipes de développement et d'exploitation est devenue complexe.

Les équipes de développement sont sous pression pour livrer toujours plus de fonctionnalités (en particulier des évolutions mineures) sur des cycles très courts, à l'aide de méthodes agiles et d'approches spécifiques telles que l'intégration en continue.

Les environnements d'exploitation ne cessent de croître, supportant une quantité extraordinaire de serveurs et de services. La production, un environnement totalement physique il y a quelques années, est presque intégralement virtualisé aujourd'hui, ce qui permet d'assembler un serveur en quelques minutes. Enfin, l'exploitation se voit dans l'obligation de superviser un volume de ressources bien plus important.

Les équipes de développement délivrent plus vite en désagrégeant des services, en parallélisant l'écriture du code et en consolidant les applications, multipliant au passage les étapes de configuration et les interdépendances. La méthode Agile implique que les développeurs passent moins de temps sur les projets internes qui, pourtant, pourraient alléger certains aspects de leur travail. D'un côté, l'exploitation rechigne à pallier les problèmes issus du développement. De l'autre, les développeurs refusent de s'investir sur des tâches de configuration et de déploiement. Au fil du temps, ces deux équipes se rejettent mutuellement la responsabilité : « Ce n'est pas le code, c'est la machine », « ce n'est pas la machine, c'est le code ». Le mouvement DevOps voit ainsi le jour, stipulant que chaque équipe doit comprendre la mission de l'autre. Cette transition impose que l'exploitation s'implique davantage dans les phases initiales du développement et qu'elle adopte des outils de gestion de configuration comme Puppet et Chef. Au-delà, le plus dur reste à faire. Ce livre blanc explique le processus d'automatisation du déploiement applicatif par l'équipe DevOps, les ressources nécessaires à ce processus et le domaine d'application (Dev, Ops ou les deux) qui les exploite sur toute la durée du cycle de vie.

Croissance de la matrice applicative, moins de maîtrise

Le cycle de vie d'une application est un processus itératif. Une application est en phase continue de développement, test, correction, mise à jour et redéveloppement en étapes distinctes : développement, test fonctionnel, test d'intégration, pré-production et production.

La terminologie et numérotation assignées à chacune de ces étapes varient en fonction de l'application et de l'entreprise. Le processus reste le même. La progression d'une étape vers une autre implique le déploiement partiel ou total de l'application vers les serveurs d'un environnement donné. Dans certains cas, il peut être nécessaire d'installer des versions antérieures ou d'en installer sur des sites parallèles (pour des besoins de test).

Pour l'équipe DevOps, la difficulté augmente à chaque ajout de composant. Toute nouvelle déclinaison d'une configuration ou d'un environnement cible entraîne la multiplication des composants, environnements et paramètres de déploiement dans la matrice. Si l'on considère le nombre de sites hébergeant le développement d'applications à intégrer, il n'est pas surprenant de constater que de nombreuses

organisations perdent la maîtrise de leurs projets. 2012 a révélé un certain nombre de défaillances majeures dans les Systèmes d'Information de très grandes entreprises. Banques, télécoms et géants des services internet s'écroulent, victimes d'évolutions non-maîtrisées ou non-anticipées. La presse informatique cite une gestion catastrophique des projets de déploiement et de mise en production pour expliquer ces échecs retentissants.

Le développement et la mise en production automatisés :

vitesse et maîtrise

L'objectif du mouvement DevOps est d'en finir avec ces problèmes. Ce mouvement est né d'une prise de conscience : les entreprises perdent la maîtrise de leurs projets informatiques précisément au point de convergence du développement et de la production. A l'origine, les équipes DevOps avaient pour mission de combler le fossé qui sépare le développement et l'exploitation en poussant les deux équipes à collaborer le plus possible pendant toute la durée du cycle de vie d'une application. C'est encore le cas aujourd'hui. Cet objectif sera désormais une des fonctions essentielles du DevOps.

La taille des environnements, plus particulièrement le nombre de serveurs requis par l'exploitation, demeure une préoccupation significative. Des produits pour la gestion de hauts volumes de configurations et de versions multiples, comme Puppet (PuppetLab) et Chef (OpsCode) sont nécessaires pour supporter les infrastructures des environnements de taille importante. Cependant, même si ces solutions sont utiles pour standardiser un parc serveur, elles sont inadaptées au support de processus applicatifs nativement multi-environnements ou multicouches. Plus simplement, les outils de gestion des configurations opèrent à la demande : c'est l'environnement et les besoins de chaque serveur qui justifient leur utilisation. Inversement, les solutions d'automatisation de mise en production utilisent des instructions. Leur utilisation s'articule autour du cycle de vie, autrement dit la définition des étapes à suivre pour promouvoir une version ou correctif d'un environnement vers un autre. Plus généralement, l'automatisation de la mise en production apporte davantage de maîtrise et de cohérence à l'ensemble des composants et, en particulier, aux applications. L'automatisation de la mise en production propose des fonctions conçues spécifiquement pour prendre intégralement en charge le cycle de vie d'une application. Ces fonctions se résument ainsi : packaging applicatif, enchaînements et modèle de déploiement.

Packaging applicatif, enchaînements et modèle de déploiement standard

Une solution d'automatisation doit être capable de supporter l'intégralité du cycle de vie d'une application avec ses différents processus organisationnels. Elle est hautement technique et propose un ensemble de fonctions exploitables aussi bien par le développement que la production. Son rôle : l'automatisation, sans effort, de toutes les étapes du déploiement d'une application en production. Le packaging applicatif est une fonction qui permet de dissocier le contenu de la logique d'exécution. Les enchaînements se substituent aux interventions manuelles et aux scripts. Enfin, le modèle de déploiement assure la gestion totale et simplifiée des paramètres d'exécution et des variables.

Le packaging applicatif

Le packaging applicatif est une fonction universelle, présente dans n'importe quelle solution d'automatisation de mise en production. Toute nouvelle version, que ce soit un correctif ou une version majeure, comprend des éléments de code provenant de sources multiples : référentiels, serveurs de développement, partage de fichiers, serveurs FTP.... Chaque composant doit contenir une référence à la version du déploiement courant. Un nouveau déploiement peut contenir des binaires, fichiers de configuration, scripts et kits d'installation ou encore des scripts de base de données. Les risques d'erreur augmentent au fur et à mesure que des éléments de code et des serveurs sont ajoutés à la matrice. Certaines fonctions doivent être automatisées, comme les tâches à répétition, l'administration des scripts et certaines exécutions. Le packaging applicatif doit pouvoir se connecter à des sources multiples et permettre aux utilisateurs de définir le contenu des packages depuis chaque source. La version et le contenu de chaque fichier sont systématiquement vérifiés et une nouvelle version du package est automatiquement générée à chaque modification (par exemple, lorsqu'une nouvelle version de base est créée). Un package représente une sorte de boitier contenant une version spécifique d'une application ou d'un service. Les packages sont des objets dynamiques. Un statut leur est assigné et peut déclencher un enchaînement. Des séquences prédéfinies garantissent qu'un package évolue progressivement et de manière maîtrisée. Enfin, les packages offrent une vue globale de l'ensemble des éléments, proposant notamment des informations sur les différents déploiements, les sites et les dates, en temps réel.

Les enchaînements

Les enchaînements réalisent le gros du travail dans une solution d'automatisation. Ils se substituent aux interventions manuelles nécessaires à l'installation, la mise à jour, l'application de correctifs ou la désinstallation et peuvent être réutilisés de façon illimitée sur l'intégralité de l'environnement d'hébergement de l'application. Ces plans sont créés à l'aide d'une interface totalement graphique. Des actions sont générées depuis une vaste bibliothèque d'actions standards et s'appliquent à la plupart des applications et des serveurs.

Les exemples ci-après montrent l'étendue des types d'infrastructure et d'outils auxquels les actions standards s'appliquent :

- Serveurs applicatifs : Weblogic, WebSphere et JBoss/IIS
- Serveurs de base de données : Oracle et SQL-Server
- Serveurs d'intégration : BizTalk
- Commandes OS, gestionnaires de package et systèmes de contrôle de sources

Les actions prédéfinies garantissent la cohérence et l'exactitude du processus de déploiement des évolutions vers les di érents serveurs. Ces actions ont un autre avantage : elles peuvent être exécutées pour chaque compte utilisateur requis (par association), en respectant les contraintes de sécurité. Enfin, leur bonne exécution est systématiquement vérifiée.

Les enchaînements proposent d'autres bénéfices :

Non-spécifiques : un enchaînement doit être totalement indépendant de l'environnement, des privilèges, des permissions et du contenu qu'il exécute. Cela permet à un plan donné d'être réutilisé pour les différentes versions de package et d'être exécuté sur n'importe quel environnement, du test à la production.

Retour arrière : la fonction de Retour arrière est essentielle pour garantir que l'application, en particulier en production, ne soit ni inactive ni bloquée, en cas de déploiement défectueux. L'ajout d'une procédure de Retour arrière à chaque enchaînement serait laborieux et inefficace : un Retour arrière après chaque étape d'une séquence alourdira considérablement le plan. Dans le cas de la solution d'automatisation Automic, chaque action prédéfinie contient une procédure de Retour arrière générique. Un Retour arrière peut être exécuté à tout moment, sans impact sur le enchaînement.

Contrôle de version : la parallélisation du développement des composants et services sur plusieurs équipes produit de nombreuses versions. Les enchaînements fonctionnent sur le même mode et il n'est pas rare que le code et les enchaînements soient développés par les mêmes équipes. Dans le cas de plusieurs équipes travaillant en parallèle, maintenir la cohérence des enchaînements et des versions passe forcément par le contrôle de version. Il est donc absolument nécessaire que votre plateforme d'automatisation autorise plusieurs équipes à collaborer sur des enchaînements de déploiement et versions multiples.

Sécurité : Les enchaînements de déploiement touchent au cœur du métier, surtout en production. Comme pour la plupart des processus informatiques, une dissociation des responsabilités est requise, parfois motivée par la règlementation en vigueur. Il arrive que le personnel responsable de la conception du plan de déploiement n'ait pas accès à l'environnement de production. Inversement, il arrive que les opérateurs responsables de l'exécution ou de la validation d'un enchaînement ne soient pas autorisés à le modifier.

Modèle de déploiement prédéfini et unifié

Beaucoup d'erreurs sont le fait de disparités dans les paramètres d'environnement sur un déploiement donné et non de fautes dans le enchaînement de déploiement. Le problème vient des logiciels et applications qui sont particulièrement sensibles aux variations (même minimes) d'environnements, comme par exemple des différences entre les répertoires racines (C:\ et E:\ sous Windows et /opt et /var sous Unix). Certaines sont plus difficiles à identifier, comme le port d'un service, les privilèges d'exécution ou la taille d'un cluster. La prise en compte de ces éléments est un véritable casse-tête. Des vérifications et réglages s'imposent pour chaque type de machine et le système. Dans le cas de centaines de serveurs à mettre à jour, une gestion mécanique de la matrice devient impossible et les scripts doivent être redéveloppés de façon systématique.

La technologie d'automatisation Automic propose un modèle permettant aux développeurs et opérateurs de définir leurs paramètres d'exécution de façon centralisée et automatique. Le modèle distribue lui-même les paramètres spécifiques à l'environnement (répertoires et versions) et les données de configuration (par exemple, les variables). Par exemple : cibles de déploiement, profils, connexions utilisateurs et plus encore. Non seulement ce type d'approche garantit la di usion correcte des paramètres au moment voulu et pour la bonne exécution, mais il simplifie la définition et la gestion des enchaînements. Sans modèle

prédéfini, l'utilisateur doit maintenir des enchaînements extrêmement complexes et stocker ses paramètres d'exécution dans des sources externes, comme du XML ou des bases de données. Ainsi le système est moins sécurisé et devient plus complexe : l'utilisateur doit lire, analyser et segmenter différents chemins d'exécution formant le enchaînement global.

En conclusion

L'automatisation de la mise en production des applications demeure la seule méthode viable pour conserver une maîtrise totale sur une matrice complexe d'applications, de versions et d'environnements. Sans méthode de contrôle et de validation des nouvelles versions, le déploiement et la mise en production provoquent des dysfonctionnements systèmes graves, de sérieux impacts financiers et rend le travail de l'équipe DevOps extrêmement périlleux. Dans le processus DevOps, l'automatisation représente une étape incontournable.

Afin de surmonter ces di icultés, il est impératif que la solution d'automatisation offre les trois fonctions suivantes :

- Le packaging applicatif et des chemins de versions, garants d'un déploiement sans erreur.
- Des enchaînements, qui offrent uniformisation, validation et cohérence du processus de déploiement.
- Des modèles de déploiement qui garantissent l'assignation des paramètres et des configurations spécifiques à chaque environnement.

Même si d'autres aspects sont à prendre en compte, une solution d'automatisation doit impérativement proposer ces trois ensembles fonctionnels pour remplir vos objectifs : rapidité, réduction des coûts et maîtrise totale.

Pour plus d'information ou pour obtenir une démonstration, veuillez consulter notre site : www.automic.com