La transition DevOps de Heartland Co-op : adoption de Git et de l'automatisation sur IBM i

Entretien avec Todd Stewart, Ingénieur Applicatif Senior certifié DevOps chez Heartland Co-op, et membre actif du **COMMON Americas Advisory Council**.

Dans cet échange franc, Todd partage les défis rencontrés et les leçons tirées lors de la conduite d'un projet de modernisation chez Heartland. Sa mission : faire évoluer 30 ans de développement sur IBM i vers une approche DevOps, faciliter l'innovation et maximiser la valeur de leur investissement IBM i.

Quel était votre objectif derrière DevOps sur IBM i ?

Todd : Nous voulions atteindre une **cohérence dans la gestion du cycle de vie applicatif** grâce à l'automatisation. En somme, éliminer les blocages dans le cycle pour un retour sur investissement rapide.
Nous avions besoin de plus de flexibilité dans le processus de changement – pour livrer des logiciels de meilleure qualité plus rapidement, et offrir des services stables, fiables et sécurisés à l'entreprise.

Notre base de code datait des années 80 et 90. Il fallait bien commencer quelque part. Un "big bang" n'était pas une option réaliste. Il fallait **automatiser notre méthode en cascade**.

Déclencheur du DevOps : une "IT à deux vitesses" – grands projets et petites corrections en parallèle

Todd : Le véritable moteur du DevOps pour nous, c'était cette "IT à deux vitesses". Nous menions de **gros projets** et des **petites corrections** en parallèle. Avant Git, ces deux lignes de développement entraient en conflit et causaient des retards.
Les petites corrections ralentissaient les grands projets, et parfois l'inverse : les priorités urgentes ne pouvaient pas être livrées à cause des conflits avec les gros chantiers.

Par exemple, nous travaillions depuis des années sur un projet consistant à changer la taille d'un attribut dans la table des localisations (passage de 2 chiffres à plus). Ce changement était énorme : **il impactait 90 % de l'ERP**. Pour le tester, nous le poussions vers l'environnement de test, ce qui **bloquait tout le reste** du développement. Impossible de tester des corrections prioritaires avec des fichiers de base de données modifiés !

Nous avons compris que le passage au DevOps serait un **changement progressif à long terme**. Il nous fallait des outils modernes.

Passer immédiatement à **ARCAD avec Git** a résolu ce problème. Git nous a permis de **détacher ce gros projet** et de libérer la voie pour les tâches quotidiennes.

Avec Git, nous pouvions **fusionner régulièrement** les petites modifications prioritaires dans le gros projet, puis finalement déployer ce projet massif dans la branche principale (**2900 modifications de code et 3700 composants déployés au total**).

Adopter Git : facile ou difficile ?

Todd : Avant ARCAD, je **n'avais aucune expérience avec Git**, même si certains membres de l'équipe l'avaient utilisé ailleurs.
Même en tant que débutant, **Git fonctionne super bien !** On peut maintenant suivre les modifications **ligne par ligne**.
Le projet ARCAD a été **un immense bénéfice pour notre département IT**. Même les développeurs IBM i les plus traditionnels sont rassurés : **peu d'interaction avec Git est nécessaire**.

Avec ARCAD, on peut **suivre les objets sans source** (comme les data areas) via Git.

J'ai appris qu'il faut **garder le dépôt propre**, éviter de laisser des branches obsolètes. Une fois qu'on déploie, on fusionne immédiatement dans la branche master et on nettoie.

Git réduit aussi le nombre d'objets "hors contrôle" – ces objets modifiés manuellement en dehors du système. Travailler en "rogue" était frustrant : leurs changements **étaient oubliés et non inclus dans les livraisons**. Git résout cela et **réintègre les développeurs dans le cycle**

Quel est le « nettoyage » nécessaire avant de passer au DevOps ?

Todd : L'**audit ARCAD** est un processus automatique qui nous a aidés à **nettoyer notre application**.
Nous avons découvert énormément de programmes et fichiers **sans source** – résultat des migrations de code au fil des années (ex. : création de vues et d'index SQL en production qui finissent sur l'environnement de dev via les refresh de données).

Nous avons réduit notre **dette technique** en supprimant des objets obsolètes, inutilisés depuis des années.

Les **références croisées dans ARCAD Observer** permettent de **tout suivre partout**, c'est génial.
Quand on modifie un fichier, **toutes les vues dépendantes sont automatiquement recompilées**.

Adopter DevOps, c'est **automatiser les tâches les plus pénibles**, pour **simplifier la vie des développeurs**.

CI/CD et automatisation des tests

Todd : Nous avons mis en place ARCAD avec **des webhooks GitHub** et des **pipelines Jenkins**.
Quand on fusionne une branche de fonctionnalité dans une branche de release, cela déclenche automatiquement un processus Jenkins qui **build une version** et l'importe dans **DROPS** pour le déploiement dans les environnements QA.

Nous prévoyons de lancer **les tests automatisés** dans ces environnements QA à l'arrivée des changements, grâce à **ARCAD Verifier**.
Il nous reste à **définir les cas de tests et les enregistrer**. Une fois cela fait, les développeurs pourront **valider leur code le soir, et retrouver les résultats de tests le**

matin.
Le **temps de déploiement va se réduire progressivement** !

Vos conseils sur Git & DevOps ?

Todd : Mon conseil c'est : **"Foncez."** Même si vous n'êtes pas encore prêts pour DevOps, les **bénéfices de Git et de l'automatisation vont bien au-delà**.

Par exemple, avec Git, vous pouvez :

- Tester un gros projet sans interférer avec le développement quotidien,
- Avoir plusieurs développeurs sur **le même composant en même temps**,
- Fusionner deux modifications dans une release unique,
- Faire monter les modifications **en parallèle dans les environnements de test**.

C'est un **gain de productivité énorme**.

Je conseille aux équipes IBM i de **mettre leur code source dans Git**.
Vous pouvez utiliser **Source Orbit d'IBM** pour migrer facilement le code depuis une bibliothèque.
Cela offre une meilleure visibilité, une **gestion des versions simplifiée**, et ARCAD for DevOps offre un **cycle CI/CT/CD très automatisé**.

Et l'avenir du DevOps sur IBM i ?

Todd : Atteindre une vraie méthodologie DevOps sur IBM i prend du temps.
Pour un DevOps complet sur un ancien code, il faudrait **moderniser une grande partie du code**, le **découper en fonctions plus petites** pour déployer plus rapidement.

Les **nouveaux outils IA** auront un rôle majeur pour aider à rendre le code plus modulaire.

Quand on **refactore du code monolithique en petits morceaux**, on peut gérer des changements parallèles dans des branches Git distinctes, ce qui **améliore considérablement la maintenance**.

Chez Heartland, nous modernisons **au cas par cas, progressivement**.
Les plus petites structures IBM i peuvent **tirer parti de Watson X et CodeAssist** pour moderniser leur base de code.

Nous prévoyons aussi d'utiliser **ARCAD Transformer Microservices** pour identifier du code redondant et le remplacer par des procédures modulaires.

Pour réussir le DevOps, il faut **les bons outils dès le départ**.
Je confirme : **la suite ARCAD est un allié précieux pour maintenir vos applications IBM i**, peu importe leur architecture !