ATLASSIAN

Outils DevOps

Choisissez des outils pour chaque phase du cycle de vie DevOps.

DevOps est la prochaine étape dans l'évolution des méthodologies Agile. Elle constitue un virage culturel qui réunit les équipes de développement et opérationnelles. Outre ce virage, DevOps est une pratique qui s'accompagne de nouveaux principes de gestion et des outils technologiques qui aident à implémenter les bonnes pratiques.

Quand il est question de chaîne d'outils DevOps, les organisations doivent rechercher des outils qui améliorent la collaboration, réduisent les changements de contexte, introduisent l'automatisation, et tirent parti de l'observabilité et de la surveillance pour livrer plus rapidement de meilleurs logiciels.

Il existe deux types principaux de chaîne d'outils DevOps : tout-en-un ou ouverte. Une chaîne d'outils DevOps tout-en-un fournit une solution complète qui ne s'intègre

Développement logiciel

Atlassian, une chaîne d'outils ouverte constitue la meilleure approche. En effet, elle peut être personnalisée avec des outils de pointe pour répondre aux besoins uniques d'une organisation. L'adoption de cette approche permet souvent d'accélérer le travail et de réduire le délai de mise sur le marché.

En savoir plus sur les chaînes d'outils DevOps.

Quel que soit le type de chaîne d'outils DevOps utilisée par une organisation, un processus DevOps doit exploiter les bons outils pour gérer les phases clés du cycle de vie DevOps :

- Découverte
- Planifier
- Build
- Test
- Surveillance
- Agir
- Feedback continu

Dans le cadre d'une chaîne d'outils DevOps ouverte, les outils sélectionnés couvrent plusieurs phases du cycle de vie DevOps. Les sections suivantes présentent certains des outils les plus populaires pour DevOps. Cela dit, compte tenu de la nature du marché, la liste évolue fréquemment. Outre les nouvelles intégrations annoncées chaque trimestre, les fournisseurs ajoutent des fonctionnalités qui leur permettent de couvrir d'autres phases du cycle de vie DevOps et, dans certains cas, consolident leurs offres pour se concentrer sur un problème spécifique rencontré par leurs utilisateurs.

Découverte


Jira Product Discovery BETA

Au cours de la phase de découverte, une équipe DevOps étudie et définit le périmètre d'un projet. Cela implique plus spécifiquement des activités telles que la recherche sur les utilisateurs, la définition des objectifs et de la réussite.

Développement logiciel

Discovery organise ces informations sous forme de données exploitables et définit la priorité des actions pour les équipes de développement. Lorsque vous définissez vos priorités, gardez également à l'esprit votre backlog de feedback des utilisateurs.

La découverte de produit est la toute première activité de conception d'un produit. C'est sur elle que s'appuiera ensuite la prise de décision. À ce stade, vous pouvez collecter toutes les informations essentielles sur les problèmes rencontrés par les utilisateurs, puis proposer des solutions.

Nous recommandons de chercher des outils qui encouragent le « brainstorming asynchrone ». Tout le monde doit pouvoir partager et commenter ce qu'il veut : idées, stratégies, objectifs, exigences, feuilles de route et documentation.

Planification

Build

Environnements de développement identiques à la production

kubernetes docker

Bien que Puppet et Chef profitent principalement aux équipes opérationnelles, les développeurs utilisent des outils open source comme Kubernetes et Docker pour provisionner des environnements de développement individuels. Par rapport aux réplications de production virtuelles et supprimables, la programmation vous permet de travailler plus efficacement.

Lorsque chaque membre de l'équipe travaille à partir d'environnements provisionnés de la même manière, le problème du « Ça fonctionne sur ma machine » cesse d'être drôle parce qu'il devient réel (comme c'est drôle).

Infrastructure-as-Code (IaC)

Développement logiciel

Les développeurs créent des apps modulaires, car elles sont plus fiables et plus faciles à administrer. Pourquoi ne pas étendre cette réflexion à l'infrastructure informatique ? Cette approche peut être difficile à appliquer aux systèmes, car ils sont en constante évolution. Nous la contournons donc en utilisant du code pour le provisionnement.

Dans l'infrastructure IaC (Infrastructure-as-Code), le reprovisionnement est non seulement plus rapide que la réparation, mais aussi plus cohérent et reproductible. En outre, vous pouvez facilement lancer des variantes de votre environnement de développement grâce à une configuration similaire à celle de la production. Le code de provisionnement peut être appliqué et réappliqué pour intégrer un serveur dans une base de référence connue. Il peut être stocké dans le système de contrôle de version. Il peut être testé, intégré dans la CI (intégration continue) et revu par des pairs.

Lorsque les connaissances institutionnelles sont codifiées dans le code, plus besoin de runbooks et de documentation interne. Des processus reproductibles et des systèmes fiables voient le jour.

Contrôle de version et programmation collaborative

Bitbucket GitHub GitLab

Il est important de contrôler les versions de votre code. Les outils dédiés permettent de stocker le code dans différentes chaînes afin que vous puissiez voir tous les changements et collaborer plus facilement en les partageant. Au lieu d'attendre la décision des conseils d'approbation des changements (CAB) avant de le déployer en production, vous pouvez améliorer la qualité et le débit du code grâce à des revues par les pairs effectuées à l'aide de pull requests.

Que sont les pull requests ? Elles informent votre équipe des changements pushés dans une branche de développement de votre dépôt. Votre équipe peut ensuite examiner les changements proposés et en discuter avant de les intégrer dans la ligne de code principale. Les pull requests améliorent la qualité des logiciels, ce qui réduit le nombre de bugs et d'incidents, limitant les coûts opérationnels et accélérant ainsi le développement.

Les outils de contrôle de version doivent s'intégrer à d'autres outils, ce qui vous permet de connecter les différentes étapes du développement et de la livraison du code. Vous

Développement logiciel

problème.

Livraison continue

L'intégration continue consiste à enregistrer du code dans un dépôt partagé plusieurs fois par jour et à le tester à chaque fois. De cette façon, vous détectez automatiquement les problèmes à l'avance, vous les corrigez au moment opportun et vous déployez le plus tôt possible de nouvelles fonctionnalités auprès de vos utilisateurs.

La revue de code par pull requests nécessite de créer une branche et est très à la mode. L'étoile Polaire de la méthodologie DevOps ? Un workflow qui permet de créer plus rapidement un nombre réduit de branches et de maintenir la rigueur des tests, sans compromettre la cadence de développement.

Recherchez des outils qui appliquent automatiquement vos tests aux branches de développement et vous donnent la possibilité de pusher vers la branche principale lorsque les builds de branche réussissent. En outre, vous obtenez un feedback continu par le biais d'alertes de chat en temps réel de la part de votre équipe grâce à une intégration simple.

Découvrez comment Bitbucket Pipelines vous aide à automatiser votre code, du test à la production.

Test

Tests automatisés

Développement logiciel

Les outils de test couvrent de nombreux besoins et fonctionnalités, y compris les tests exploratoires, la gestion des tests et l'orchestration. Toutefois, pour la chaîne d'outils DevOps, l'automatisation est une fonction essentielle. Les tests automatisés sont payants au fil du temps, puisqu'ils accélèrent vos cycles de développement et de test sur le long terme. Dans un environnement DevOps, ils sont également importants pour favoriser la sensibilisation.

Si elle est effectuée tôt et régulièrement, l'automatisation des tests permet d'améliorer la qualité logicielle et de réduire les risques. Les équipes de développement peuvent exécuter à plusieurs reprises des tests automatisés qui couvrent plusieurs domaines, notamment l'interface utilisateur, la sécurité ou la charge. Elles peuvent aussi obtenir des rapports et des graphiques de tendance qui aident à identifier les domaines à risque.

Les risques sont indissociables du développement de logiciels, mais vous devez d'abord les anticiper pour pouvoir les limiter. Faites une faveur à votre équipe opérationnelle et aidez-la à comprendre les rouages. Recherchez des outils compatibles avec les wallboards et laissez toutes les personnes impliquées dans le projet commenter les résultats de build ou de déploiement spécifiques. Essayez de privilégier les outils qui facilitent la participation des équipes opérationnelles aux tests rapides et exploratoires.

Déploiement

Tableaux de bord des déploiements

Réunir toutes les informations sur les changements, les tests et le déploiement en prévision d'une livraison en un seul et même endroit constitue l'un des aspects les plus

Développement logiciel

qu'interviennent les tableaux de bord de livraison.

Recherchez des outils avec un tableau de bord unique intégré à votre dépôt de code et à vos outils de déploiement. Trouvez-en un qui vous permet de visualiser l'ensemble des branches, des builds, des pull requests et des avertissements de déploiement en un seul et même endroit.

Déploiement automatisé

Bitbucket AWS CodePipeline

Il n'y a pas de recette miracle pour garantir le fonctionnement d'un déploiement automatisé sur l'ensemble des apps et des environnements informatiques. Mais la conversion du runbook des opérations en script exécutable via l'invite de commandes à l'aide de Ruby ou de bash est un bon point de départ. Les bonnes pratiques d'ingénierie sont essentielles. Utilisez des variables pour exclure les noms d'hôte. Garantir des scripts ou du code uniques pour chaque environnement n'est pas amusant (et passe à côté de l'objectif visé). Créez des scripts ou des méthodes utilitaires pour éviter toute duplication du code. Faites évaluer vos scripts par des pairs pour contrôler leur intégrité.

Essayez d'abord d'automatiser les déploiements dans votre environnement de niveau le plus bas (où vous utiliserez cette automatisation le plus fréquemment), puis répliquez le processus jusqu'à la production. Cet exercice permet ni plus ni moins de mettre en évidence les différences entre vos environnements et de générer une liste de tâches pour les standardiser. En prime, la standardisation des déploiements via l'automatisation réduit la « dérive des serveurs » au sein des environnements et entre eux.

Agir

Suivi des incidents, des changements et des problèmes

Jira Service Management Jira Software Opsgenie

Développement logiciel

Pour permettre la collaboration entre les équipes DevOps, ces dernières doivent impérativement visualiser le même travail. Que se passe-t-il lorsque des incidents sont signalés ? Sont-ils traçables et liés à des problèmes logiciels ? Les changements apportés concernent-ils les livraisons ?

Rien n'entrave davantage la collaboration entre les équipes de développement et opérationnelles que les incidents et les projets de développement logiciels suivis dans différents systèmes. Recherchez des outils qui centralisent les incidents, les changements, les problèmes et les projets de développement sur une plateforme de sorte à pouvoir identifier et résoudre les problèmes plus rapidement.

Observation

Surveillance des performances des apps et des serveurs

Deux types de surveillance doivent être automatisés : la surveillance des performances des apps et des serveurs.

Vous pouvez « cocher » une case ou tester votre API manuellement pour une vérification ponctuelle. Mais pour comprendre les tendances et l'intégrité générale de votre app (et de vos environnements), vous avez besoin d'un logiciel qui capture et enregistre les données 24 h/24, 7 j/7. L'observabilité continue est une fonctionnalité clé pour les équipes DevOps performantes.

Recherchez des outils qui s'intègrent à votre client de discussion de groupe afin que les alertes soient envoyées directement dans la discussion de votre équipe ou dans une

Développement logiciel

Feedback continu

Il vous suffit d'écouter les clients pour savoir si vous avez conçu la solution adéquate. Le feedback continu inclut à la fois la culture et les processus à suivre pour recueillir régulièrement du feedback, ainsi que des outils permettant de tirer les conclusions de ce dernier. Les pratiques de feedback continu incluent la collecte et l'examen des données NPS, des enquêtes sur la perte de clients, des rapports de bug, des tickets de support et même des tweets. Dans une culture DevOps, tous les membres de l'équipe produit ont accès aux commentaires des utilisateurs, car ils orientent toutes les tâches, de la planification des livraisons aux sessions de tests exploratoires.

Recherchez des apps qui intègrent votre outil de discussion à votre plateforme d'enquête préférée pour obtenir un feedback de type NPS. Twitter et/ou Facebook peuvent également être intégrés à la discussion pour bénéficier d'un feedback en temps réel. Pour approfondir le feedback provenant des réseaux sociaux, il s'avère judicieux d'investir dans une plateforme de gestion de ces derniers capable d'élaborer des rapports sur la base des données historiques.

L'analyse et l'intégration du feedback donnent parfois l'impression de ralentir le rythme de développement à court terme, mais elles sont plus efficaces à long terme que la livraison de nouvelles fonctionnalités dont personne ne veut.

Conclusion...

Chez Atlassian, nous croyons qu'il est essentiel de disposer d'une chaîne d'outils DevOps qui s'intègre aux outils que les équipes de développement et opérationnelles adorent utiliser. C'est pourquoi nous avons développé une plateforme DevOps qui s'intègre à plus de 171 fournisseurs tiers de premier plan. Vous pouvez ainsi prendre les

Développement logiciel

Pour commencer, [essayez gratuitement la solution DevOps d'Atlassian](#).

ARTICLE SUIVANT

Considérations relatives à votre chaîne d'outils DevOps →

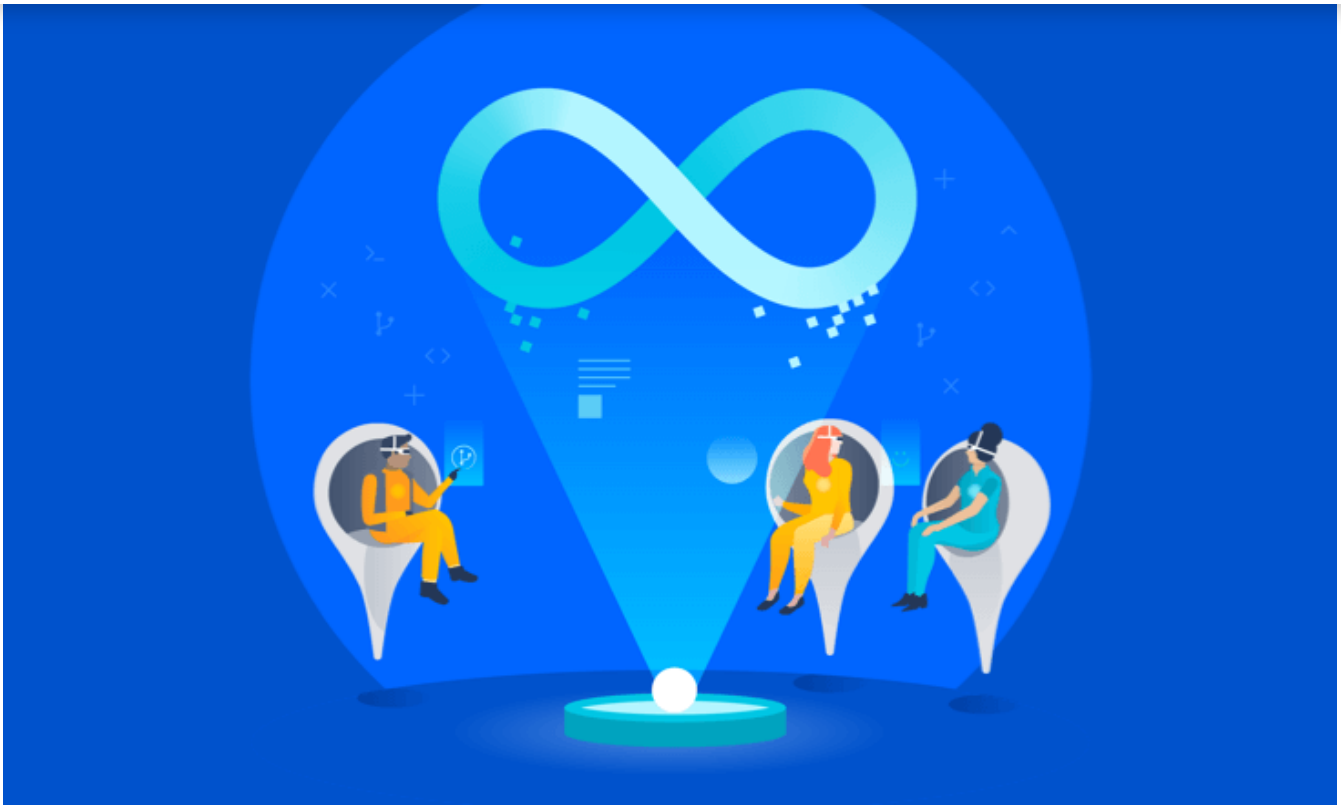
Lectures recommandées

Ajoutez ces ressources à vos favoris pour en savoir plus sur les types d'équipes DevOps, ou pour les mises à jour continues de DevOps chez Atlassian.

Communauté DevOps

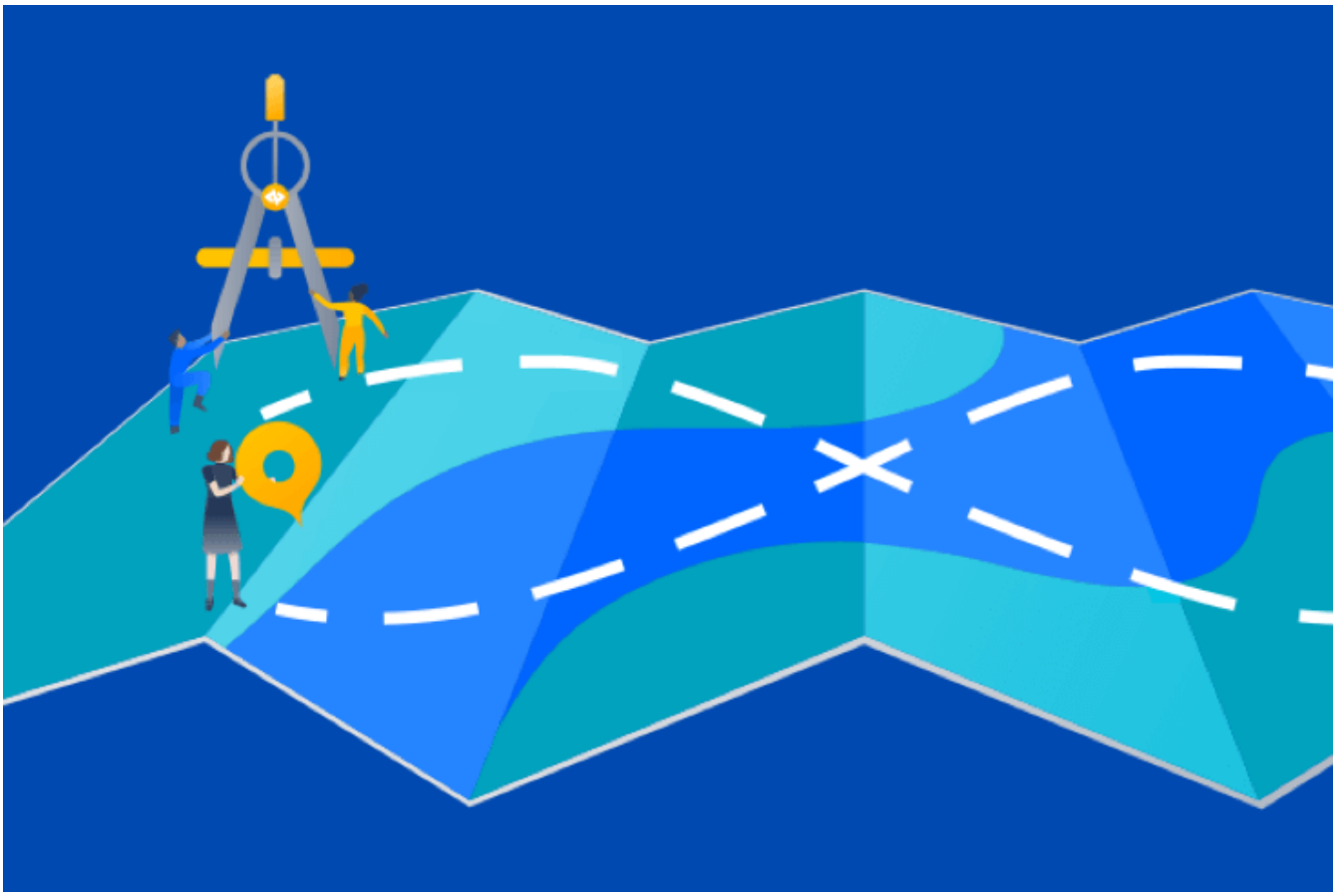
En savoir plus →

Développement logiciel



Parcours de formation DevOps

En savoir plus →



Développement logiciel

Inscrivez-vous à notre newsletter Devops

Adresse e-mail

Inscrivez-vous

ATLASSIAN

PRODUITS

Jira Software
Jira Align
Jira Service Management
Jira Work Management
Jira Product Discovery
Confluence
Trello
Bitbucket

Consulter tous les produits

RESSOURCES

Support technique
Achats et licences
Communauté Atlassian
Base de connaissances
Marketplace
Mon compte

Créer un ticket de support

DÉVELOPPER ET APPRENDRE

Partenaires
Formation et certification
Documentation
Ressources développeurs
Services Enterprise

Afficher toutes les ressources

À PROPOS D'ATLASSIAN

Entreprise
Carrières
Événements
Blogs

Développement logiciel

Confiance et sécurité

Nous contacter

Français ▼

Politique de confidentialité

Vos choix en matière de confidentialité

Conditions

Mentions légales