Jenkins: C'est un outil d'automatisation d'intégration continue open-source qui permet de compiler, tester et déployer des projets de manière continue. Jenkins offre une grande flexibilité et de nombreuses fonctionnalités grâce à ses plugins.

Nexus: C'est un référentiel de gestion de packages qui permet de stocker et de gérer les artefacts de votre projet, tels que les bibliothèques, les binaires et les fichiers de configuration. Nexus peut être utilisé pour stocker les artefacts générés lors des builds Jenkins.

Git: C'est un système de contrôle de version distribué qui permet de suivre les changements apportés aux fichiers de code source et de les gérer au fil du temps. Git facilite le travail collaboratif sur des projets et permet de revenir facilement à des versions antérieures si nécessaire.

GitLab: C'est une plateforme de gestion de projets qui intègre des fonctionnalités de contrôle de version, de gestion de problèmes, d'automatisation de la livraison et de collaboration. GitLab est conçu pour permettre aux équipes de travailler plus efficacement sur des projets en offrant une interface utilisateur conviviale et une intégration étroite avec Git.

En général, les outils que vous avez sélectionnés devraient suffire pour mettre en place l'automatisation de votre processus de développement et de déploiement de données. Cependant, en fonction des besoins spécifiques de votre projet, il est possible que d'autres outils soient nécessaires, tels que des outils de gestion de la qualité du code, des outils de test automatisé ou des outils de surveillance des performances.

1. Développement : Cette étape consiste à écrire et à modifier le code source de l'application. Les développeurs travaillent généralement sur leur propre machine ou leur propre branche Git.
2. Intégration continue (CI) : Lorsqu'un développeur soumet un nouveau code source, l'outil d'intégration continue (Jenkins) déclenche automatiquement une série d'actions. Tout d'abord, il compile le code source pour s'assurer qu'il est valide. Ensuite, il exécute des tests unitaires et d'intégration pour s'assurer que le code ne provoque pas de régression. Enfin, il génère un package contenant le code compilé et testé.
3. Stockage de packages : Le package généré par l'outil d'intégration continue est stocké dans le référentiel de packages (Nexus). Le package est disponible pour les étapes suivantes du processus d'automatisation.
4. Déploiement continu (CD) : L'outil de déploiement continu (Jenkins) récupère le package généré par l'outil d'intégration continue à partir du référentiel de packages et le déploie automatiquement dans l'environnement de production.
5. Surveillance : L'environnement de production est étroitement surveillé pour détecter tout problème pouvant survenir lors du déploiement. Si des problèmes sont détectés, ils sont rapidement signalés à l'équipe de développement pour qu'ils puissent être résolus.
6. Répétition : Ce processus se répète à chaque fois qu'un développeur apporte des modifications au code source. Les modifications sont intégrées continuellement et déployées automatiquement dans l'environnement de production.

Si les développeurs travaillent sur Informatica, cela ne change pas fondamentalement l'utilisation de GitLab dans l'architecture logique. GitLab serait toujours utilisé pour stocker le code source de l'application, gérer les demandes d'extraction, configurer des pipelines d'intégration continue et de déploiement continu, et pour intégrer avec Jenkins.

Cependant, il est possible que le processus de développement spécifique pour Informatica soit différent de celui pour d'autres types d'applications. Par exemple, vous devrez peut-être configurer GitLab pour stocker les projets Informatica ou les workflows de données plutôt que le code source traditionnel. De plus, vous devrez peut-être intégrer des outils Informatica spécifiques dans les pipelines d'intégration continue et de déploiement continu.

Il est donc important de comprendre comment les développeurs travaillent avec Informatica et de personnaliser l'utilisation de GitLab et des autres outils en conséquence.

Dans le contexte de votre projet, l'application Informatica est utilisée pour la création et la gestion des flux de données. Ainsi, la première étape consiste à mettre en place un système de gestion de code source basé sur Git et GitLab pour stocker le code source des développements Informatica. Ce code source est ensuite intégré dans un environnement d'intégration continue basé sur Jenkins. Les modifications apportées au code source sont vérifiées automatiquement en utilisant des tests d'intégration continue avant de permettre une fusion en continu du code source.

Le serveur Nexus est utilisé pour stocker les packages des dépendances des applications. L'application Informatica nécessite des dépendances externes qui doivent être stockées dans Nexus pour faciliter le déploiement et la gestion de ces dépendances.

Enfin, l'étape de déploiement utilise Jenkins pour orchestrer le déploiement de l'application sur les différents environnements, y compris l'environnement de production. La livraison continue de l'application permet une mise à jour régulière des différentes versions d'application tout en assurant une qualité constante.

Ainsi, pour votre projet, la première étape consisterait à mettre en place un système de gestion de code source basé sur Git et GitLab pour stocker le code source des développements de modèles de données et de métadonnées. Le code source est ensuite intégré dans un environnement d'intégration continue basé sur Jenkins, où il sera vérifié automatiquement à l'aide de tests d'intégration continue avant de permettre une fusion en continu du code source.

Le serveur Nexus est utilisé pour stocker les packages des dépendances des applications, y compris les dépendances d'Informatica, pour faciliter le déploiement et la gestion de ces dépendances.

Enfin, l'étape de déploiement utilise Jenkins pour orchestrer le déploiement des modèles de données et des métadonnées sur les différents environnements, y compris l'environnement de production. La livraison continue permet une mise à jour régulière des différents éléments tout en assurant une qualité constante.

|  |  |  |  |
| --- | --- | --- | --- |
| Critère | Ansible | CircleCi | Jenkins |
| Déploiement | Gestion de la configuration et déploiement | Déploiement continu | Intégration continue et déploiement continu |
| Plugins et Extensions | Bibliothèque de modules ansible | Bibliothèque de plugins Circileci | Large choix de plugins et d’extensions |
| Support multiplateforme | Windows, Linux MacOs | Windows, Linux MacOs | Windows, Linux MacOs |
| Performance | Moyen avec une faible empreinte mémoire | Rapide | Rapide |
| Satisfaction accrue des clients | Facilité de la mise en production des applications | Améliore la qualité de livraison | Amélioration et réduction de la mise en production |
| Stabilité et fiabilité de système | Architecture simple | Plateforme cloud fiable | Plateforme mature et stable , utilisé depuis des années |
| Cout | Gratuit et open source | Abonnement mensuel | Gratuit et open source |

|  |  |  |  |
| --- | --- | --- | --- |
| Critère | Nexus Repository Manager OSS | Artifactory de Jfrog | Aws CodeArtifact |
| Intégration | Intégration avec divers outils Populaires | Intégration avec divers outils Populaires | Intégration avec d’autres services AWS |
| Performance | Performance élevée pour les gros dépôts de code source | Performance élevée pour les gros dépôts de code source | Performance élevée pour les gros dépôts de code source |
| Support multiplateforme | Multi-plateforme | Multi-plateforme | AWS |
| Configuration | Configuration simplifié flexible | Configuration flexible | Configuration simple avec l’interface AWS |
| Intégration | Forte avec plusieurs outils | Moyenne | Faible |
| Stabilité et fiabilité de système | Mature et fiable et le plus utilisé dans le monde | Mature et stable | Mature et stable |
| Cout | Gratuit pour la version open source | Gratuit pour la version open source mais très limité et Tarifs en besoin des fonctions | Tarifs basés sur l’utilisation |

|  |  |  |
| --- | --- | --- |
| KDD : Knowledge Discovery in Database​ | Sample, Explore, Modify, Model, Assess​ | Cross Industry Standard Process for Data Mining​ |
| ---------- | ---------- | Business Understanding​ |
| Sélection | Sample | Data Understanding​ |
| Preprocessing | Explore​ | Data Understanding |
| Transformation | Modify​ |
| Data Mining​ | Model​ | Modeling |
| Interpretation/Evaluation​ | Assess​ | Evaluation |
| --------- | --------- | Deployment |

1. Agile Scrum : La méthodologie Agile Scrum peut être utilisée pour les projets DataOps qui nécessitent des livraisons fréquentes et itératives. Les équipes de projet peuvent travailler sur des sprints courts de deux à quatre semaines pour livrer rapidement des résultats à valeur ajoutée. Les principes de collaboration et de flexibilité de la méthodologie Agile Scrum sont particulièrement adaptés à la nature évolutive de l'environnement DataOps.

Scrum peut être préférable si :

* Le projet de dataops a des objectifs clairs et précis à atteindre à la fin de chaque sprint.
* L'équipe a besoin de définir des rôles et des responsabilités claires.
* L'équipe souhaite travailler en sprint avec des objectifs clairs à atteindre.
* L'équipe a besoin d'une approche plus structurée et planifiée pour mener à bien le projet.

Kanban peut être préférable si :

* Le projet de dataops nécessite une grande flexibilité.
* L'équipe souhaite se concentrer sur la livraison continue et la réduction des temps de cycle.
* Le projet implique un flux continu de tâches et de demandes.
* L'équipe a besoin de visualiser et de gérer efficacement l'état d'avancement du projet.
* Le projet implique une collaboration étroite avec les parties prenantes.

DataOps est une approche plutôt qu'une méthodologie spécifique. C'est une approche qui vise à améliorer l'efficacité, la qualité et la sécurité des flux de travail de données tout en favorisant la collaboration et l'automatisation.

DataOps se concentre sur l'ensemble du cycle de vie des données, de la collecte à la livraison, en passant par la préparation, le stockage, l'analyse et la visualisation. Il encourage l'utilisation de pratiques DevOps pour améliorer la qualité, la sécurité et la rapidité de la livraison des solutions de données.

Bien qu'il n'y ait pas une méthodologie DataOps standardisée, il existe plusieurs frameworks et outils qui peuvent aider à appliquer les principes de DataOps à différents projets de données. Ces frameworks et outils peuvent être utilisés pour structurer et organiser les différents processus de DataOps, tels que la collecte, la transformation et l'analyse des données, la gestion des versions et des tests, la surveillance et la maintenance des environnements de données.