Comment monter concrètement une usine logicielle moderne ?

Monter une usine logicielle moderne c'est avant tout se poser la question du niveau d'intégration au regard de l'ambition de son entreprise et définir deux piliers essentiels : la gestion du contrôle de code source (SCM) et les pratiques combinées d'intégration et de livraison continue (CI/CD).

Définir son système de gestion du contrôle de code source

Le SCM va permettre aux développeurs de versionner le code et collaborer sur un même projet en partageant un arbre de code commun. La collaboration des développeurs peut se matérialiser par des revues de code, pouvant être en partie automatisée par des outils d'analyse structurelle afin d'obtenir un code propre et agnostique, mais également d'optimiser le temps de revue des développeurs.

Il est possible d'opter pour une solution de SCM en SaaS ou hébergée sur site. Outre l'aspect financier, le choix sera conditionné à des questions de sécurité, d'architecture d'entreprise ou d'infrastructures disponibles entre autres. Github et Gitlab, aujourd'hui leader du marché, proposent les deux modèles avec une approche néanmoins différente.

GitLab est spécialisé dans une solution clé en main où tous les aspects de l'usine logicielle sont intégrés. De son côté, GitHub propose une approche plus spécialiste et centrée sur le SCM, en offrant une grande flexibilité d'intégration pour des outils tiers. Ce positionnement est aujourd'hui en train d'évoluer et de se rapprocher, suite notamment à son rachat par Microsoft, qui pousse l'ajout de fonctionnalités clé en main comme GitHub Actions soutenu par son cloud Azure.

Définir ses pratiques d'intégration et de livraison continue

L'un des objectifs d'une usine logicielle est d'optimiser les flux de production et de livraison tout en garantissant la qualité. C'est ici qu'interviennent les outils de CI/CD de l'usine (ce que l'on appelle parfois du DevOps) qui permettent de tester, valider et déployer de façon continue les nouveaux incréments de code réalisés par les équipes.

Cette intégration continue est permise grâce à l'exécution automatique d'outils d'analyse fonctionnelle garantissant le bon fonctionnement du code. Ainsi, tout ajout au projet est validé automatiquement via des tests unitaires ou fonctionnels, ce qui limite les risques d'introduire des bugs dans l'application. L'intégration continue peut ensuite donner lieu au déploiement continue pour automatiser les déploiements et la mise à disposition des applications une fois les standards de qualités atteints.

Dans une approche agile et qualitative, cette automatisation permet d'éviter les déploiements manuels, les erreurs humaines afin de réallouer ce temps sur des tâches à plus haute valeur ajoutée. In fine, l'entreprise recevra plus rapidement les appréciations de ces utilisateurs finaux et pourra réagir plus vite en conséquence pour faire des ajustements.

Les outils de CI/CD sont également disponibles en Saas ou sur site et, contrairement au SCM, son facteur coût doit être pris en compte puisqu'ils vont être consommateurs de beaucoup plus de ressources informatiques (réplique du travail d'un développeur, téléchargement du code source, besoin en réseau, CPU, ram pour composer et exécuter le code). C'est un véritable choix stratégique à réfléchir, souvent en fonction de la taille de l'entreprise et de l'anticipation de sa croissance future.

Pour une startup de moins de 50 employés, il peut être intéressant d'adopter un outil spécialisé CI/CD as a service qui permettra de limiter le coût de la maintenance et la mise en place que l'on viendra brancher à l'outil de SCM.

Les leaders de ces outils sont aujourd'hui CircleCI, TravisCI et Bitrise (spécialisé pour application mobile pour ce dernier). Des outils sur site sont également disponibles comme Jenkins. GitHub et GitLab intègrent quant à eux un service de CI/CD hybride permettant de mixer un usage sur site ou Saas, adapté pour une approche scalable, idéale pour des entreprises plus grandes.

Si nous prenons l'exemple de Fabernovel, qui développe à la fois des applications iOS et Android, sa CI/CD pour des projets iOS nécessite l'usage de machines macOS dont la disponibilité via le cloud est pratiquement impossible ou à un prix élevé (par exemple MacStadium). C'est pourquoi il a été plutôt décidé d'utiliser des machines sur site pour iOS.

En revanche, sans contrainte particulière pour Android, Fabernovel utilise des machines dans le cloud afin d'avoir des coûts et des disponibilités directement liés aux usages (scalabilité).
GitHub Actions permet cette hybridation des CI/CD. Pour entrer plus dans les détails techniques, nous avons publié en open source la manière dont les équipes utilisent le cloud de Google pour exécuter ces CI/CD de la majorité des projets Android.

D'autres usages comme l'utilisation de matériel spécialisé non disponible dans le cloud peuvent également motiver des approches sur sites, comme par exemple le besoin en GPU dans le cas de machines learning ou data mining.

Nouvelles tendances, nouveaux usages aux Etats-Unis

Aux Etats-Unis, la tendance est à l'utilisation d'outils d'usine logicielle qui peuvent être configurés par du code : par exemple Github Actions est un outil de CI/CD as Code. De tels outils permettent un dialogue efficace entre équipe de développement et équipe d'infrastructure et facilite donc les approches DevOps.

Par exemple, il est possible de venir directement coder les étapes du déploiement dans le code source du projet. Par ailleurs, ces informations, présentées sous forme de code source, peuvent

être versionnées et soumises également à des analyses structurelles et fonctionnelles mises en place par l'usine logicielle.

L'usine logicielle est un outil essentiel à la performance et à la qualité des projets. Cette tendance "as code" s'inspire d'un mouvement plus général initié par l'usage de technologies de plus en plus complexes dans le monde de l'infrastructure, en particulier du côté des cloud providers et qui permet de les modéliser et les rationaliser.