Serverless computing : entre mode et promesse ?

L'informatique sans serveur (le « serverless ») vient alimenter cette tendance et, comme pour le XaaS (« Everything as a service »), permet de sous-traiter la difficile gestion de l'infrastructure à des fournisseurs cloud. Les ressources consacrées au déploiement, à la gestion de la capacité et à la sécurisation de l'infrastructure peuvent désormais être redirigées et investies directement dans le développement des applications elles-mêmes.

Le serverless ouvre des opportunités prometteuses d'innovation, de capacité et d'évolutivité, mais qui dépendent du contexte et des objectifs propres à chaque entreprise.
En effet, si le serverless permet de réduire la complexité sur certains aspects, il introduit de nouvelles problématiques par ailleurs. L'absence actuelle de standards ouverts de compatibilité entre plateformes et la dépendance relative envers le fournisseur en est une. C'est pourquoi cette technologie à fort potentiel doit être bien comprise et appliquée de façon précise aux objectifs de l'entreprise.

Se concentrer sur le code, pas sur les serveurs

Au-delà des services PaaS du cloud, le serverless est un modèle d'architecture logicielle qui permet aux développeurs de déployer des composants applicatifs modulaires et autonomes sous forme de fonctions ou de containers et de déléguer au fournisseur l'intégralité de l'infrastructure et du middleware, de son opération, de l'exécution du code et de l'allocation dynamique des ressources en fonction de la charge.

Le serverless présente plusieurs intérêts pour les équipes applicatives. Il offre une grande facilité de mise en œuvre et permet une grande vélocité, du codage au déploiement en production, avec une autonomie maximale des développeurs. La facturation variable reflète l'exacte durée de l'utilisation des ressources et permet de s'affranchir de coûts fixes de l'infrastructure et des compétences nécessaires à son cycle de vie. Le serverless offre aussi une gestion automatique de la capacité à très grande échelle, permettant d'accommoder sans effort ni planification de grandes variations de charge.

Ce dernier est aujourd'hui très utilisé pour le développement rapide de code avec un temps d'exécution court tel que des microservices Web, la transformation de données (ETL), la programmation par événement, mais aussi l'orchestration et l'automatisation des processus IT et applicatifs dans le cloud. Le modèle se prête bien à des usages sporadiques, ou connaissant de très grandes variations de volume rendant le dimensionnement au pic coûteux et difficile à planifier. Si l'usage du serverless augmente rapidement dans ces usages périphériques, on observe encore peu d'architectures exploitant cette technologie au cœur de la logique de leurs applications.

De nouvelles formes de complexité

Si le serverless simplifie les problématiques liées à l'infrastructure et accroît l'agilité des développeurs, il augmente en contrepartie la complexité à d'autres niveaux, comme le risque de fragmentation accrue de l'architecture applicative. Individuellement, chacune des fonctions s'en trouve simplifiée et le cycle de développement accéléré, mais le nombre accru de dépendances, d'interfaces et d'appels entre fonctions rend également la structure globale du système sensiblement plus complexe. Les efforts de configuration sont plus importants et les scripts de déploiement plus dispersés dans les équipes. En conséquence, le développement de logiciels est de facto plus efficace ou agile, mais la maîtrise de l'architecture et le suivi du comportement et de la performance des applications en production s'en trouve complexifié.

Comprendre et maîtriser les variations de performance

Un des avantages du serverless est son évolutivité. Que l'on ait quelques centaines ou des milliards de demandes à traiter, le système se met à l'échelle automatiquement. En revanche, les performances des requêtes individuelles peuvent être moins prévisibles : une requête peut s'exécuter en quelques millisecondes comme cent fois plus selon l'état du système, en particulier lors des reprises d'activité après une période sans invocation, nécessitant une allocation initiale de ressources (« cold start »). Ces variations de performance à bas volume doivent être mesurées, comprises et mitigées afin d'éviter des dégradations significatives de performance de l'application, voire son effondrement.

Heureusement, la prédictibilité des performances du serverless augmente généralement avec la charge. A faible volume, les résultats apparaissent parfois peu concluants et contre-intuitifs, révélant des performances aberrantes, mais lorsque le nombre d'invocations s'accroît sensiblement, l'allocation moins erratique des ressources, et la mise en cache des données accédées le plus fréquemment, améliorent fortement et stabilisent la performance. La contrepartie devient alors la complexité, la fragmentation et la volatilité de l'architecture globale et la maîtrise de sa performance sous forte charge.

L'adoption du serverless doit donc s'accompagner d'une réelle discipline des développeurs en matière d'observabilité et de télémétrie logicielle, et l'instrumentation du code doit faire partie des responsabilités des équipes DevOps.
Même si son utilisation peut initialement apparaître comme périphérique et non critique, il est essentiel d'intégrer le code serverless dans la stratégie globale de monitoring et d'observabilité applicative. Cela permet d'assurer la disponibilité, la fiabilité et la performance des microservices et de l'application à tous les stades du développement et de la croissance de l'activité.

Coller aux exigences métier

Une architecture serverless repose sur un principe de praticité technique fondé sur l'autonomie, l'accessibilité, la flexibilité et la variabilité, qui attire naturellement les développeurs. En contrepoint, la question de sa finalité ne doit donc pas être mise de côté. L'objectif est-il de développer une application complexe basée uniquement sur des services serverless ? Ou le recours au Serverless doit-il être limité à certains fonctions ciblées où il excelle ? Et si oui, lesquels ? Comment le serverless influence-t-il l'architecture applicative et les processus de développement et d'opération ? Comment préserver la cohérence des services applicatifs et leur performance, en adéquation avec la stratégie produit ?

Il est important de ne pas considérer le concept du serverless comme une décision purement technique, mais plutôt le voir comme un des facteurs d'accélération et de transformation de l'entreprise et de son architecture de services. Cela commence avec les équipes de développement, qui doivent continuer d'évoluer de leur rôle de codeur vers celui de concepteurs d'expériences clients d'excellence, et axés sur la résolution de problèmes au service des utilisateurs et de la performance business de l'entreprise.