



on est responsable de provisionner et de manager leurs ressources, ce qui peut poser plusieurs problèmes.

- On doit maintenir les serveurs disponibles même lorsqu'il n'y a pas de requêtes à traiter,
- On est responsable de la disponibilité et de la maintenance des serveurs et de leurs ressources,
- On doit ajuster les serveurs avec la charge : augmenter lorsque la charge arrive et diminuer lorsque la charge redescend.

Cela peut être très difficile à gérer pour les développeurs individuels. Cela finit par nous éloigner de notre mission initiale : construire et maintenir des applications au quotidien. Cela relève le plus souvent de la responsabilité de l'équipe infrastructure et rarement des développeurs. Cependant, les processus nécessaires pour les supporter peuvent ralentir les développements. On ne peut pas développer d'application sans l'aide de l'équipe infrastructure. En tant que développeurs, on recherche une solution à ces problèmes et c'est là que le Serverless entre en jeu.

#### — **L'architecture Serverless**

L'architecture Serverless est un modèle dans lequel le fournisseur de services Cloud (AWS, Azure ou Google Cloud) est responsable de l'exécution d'un morceau de code en allouant de manière dynamique les ressources. Et il ne facture que la quantité de ressources utilisées pour exécuter le code. Le code est généralement exécuté dans des conteneurs sans état pouvant être déclenchés par divers événements, notamment des requêtes HTTP, des événements de base de données, des services de file d'attente, des alertes de surveillance, des téléchargements de fichiers, des événements planifiés, etc. Le code envoyé au fournisseur de Cloud pour l'exécution est généralement sous la forme d'une fonction. Par conséquent, Serverless est parfois appelé «Functions as a Service» ou «FaaS». Alors que le Serverless isole l'infrastructure sous-jacente du développeur, les serveurs sont toujours impliqués dans l'exécution de nos fonctions. Étant donné que notre code va être exécuté en tant que fonctions individuelles, nous devons être conscients de certaines choses.

#### — **Microservices**

Le plus grand changement auquel on est confrontés lors de la transition vers un monde Serverless est que notre application doit être structurée sous forme de fonctions. Nous avons