



# Cloud Computing

– Cours 4 –

## Chapitre 4 : Le Calcul sans Serveur

**Dr. MENNOUR R.**

Faculté des nouvelles technologies

rostom.mennour@univ-constantine2.dz

### Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	Master 1	SDIA

# Plan du cours

- **Comment en est on arrivé là ?**
- **Qu'est ce que le calcul sans serveur ?**
- **Comment le calcul sans serveur fonctionne**
- **Comment c'est différent ?**
- **Avantages et limites**



# **Section 1 :**

## **Comment en est-on arrivé là ?**

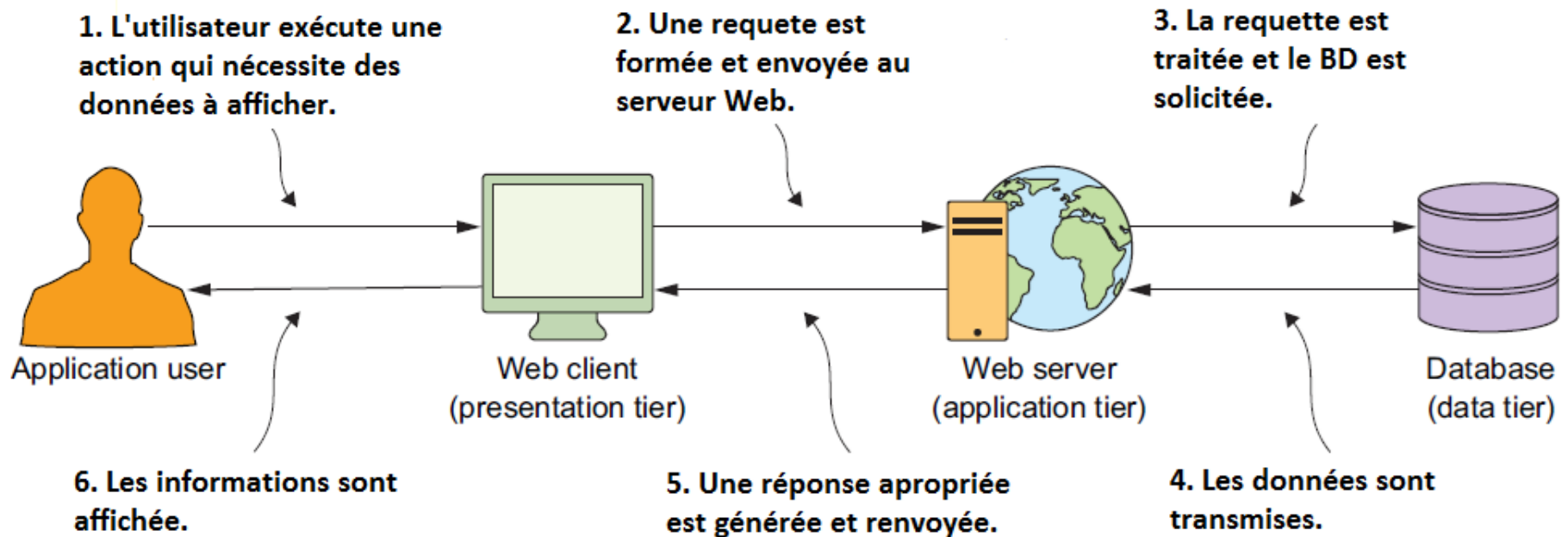
# Comment en est-on arrivé là ?

## Les architectures classiques

- Si vous regardez les systèmes qui alimentent la plupart des logiciels Web d'aujourd'hui, vous verrez des serveurs principaux effectuer diverses formes de calcul et des frontaux côté client offrant une interface permettant aux utilisateurs de fonctionner via leur navigateur, leur mobile ou leur ordinateur de bureau.
- Dans une application Web classique, le serveur accepte les demandes HTTP du frontal et traite les demandes. Les données peuvent traverser de nombreuses couches d'application avant d'être enregistrées dans une base de données.
- Naturellement, la plupart des systèmes sont plus complexes une fois que des éléments tels que l'équilibrage de charge, les transactions, le clustering, la mise en cache, la messagerie et la redondance des données sont pris en compte.
- La plupart de ces logiciels nécessitent des serveurs exécutés dans des centres de données ou dans le cloud qui doivent être gérés, maintenus, corrigés et sauvegardés.
- Le provisionnement, la gestion et la correction des serveurs est une tâche qui prend du temps et qui nécessite souvent des personnes dédiées aux opérations. Un environnement non trivial est difficile à mettre en place et à fonctionner efficacement. L'infrastructure et le matériel sont des composants nécessaires de tout système informatique, mais ils sont souvent aussi une distraction par rapport à ce qui devrait être l'objectif principal: résoudre le problème commercial.

# Comment en est-on arrivé là ?

## Les architectures classiques



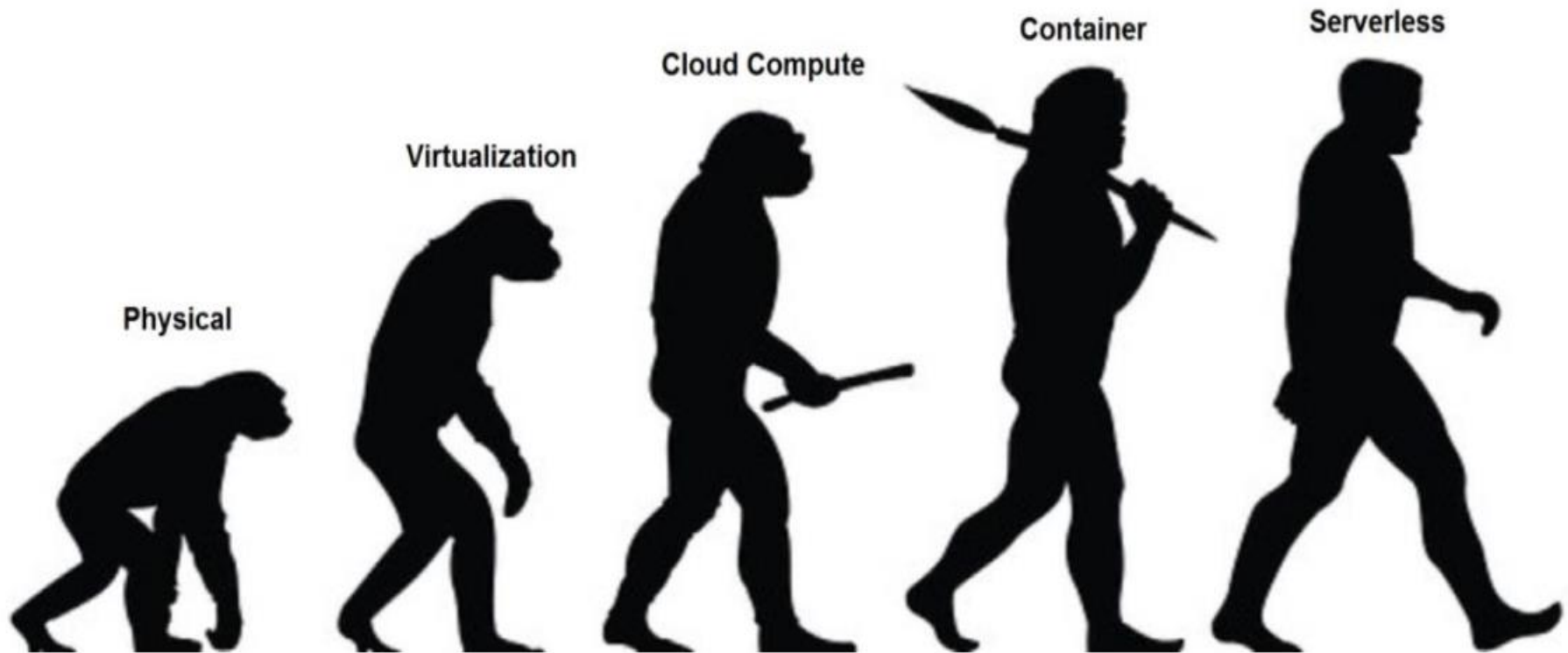
# Comment en est-on arrivé là ?

## Les architectures orientées service et les microservices

- Parmi les architectures de systèmes et d'applications, l'architecture orientée services (SOA) a beaucoup de reconnaissance parmi les développeurs de logiciels. C'est une architecture qui conceptualise clairement l'idée qu'un système peut être composé de nombreux services indépendants. Beaucoup de choses ont été écrites sur SOA, mais elles restent controversées et mal comprises car les développeurs confondent souvent la philosophie de conception avec une implémentation et des attributs spécifiques.
- SOA ne dicte pas l'utilisation d'une technologie particulière. Au lieu de cela, il encourage une approche architecturale dans laquelle les développeurs créent des services autonomes qui communiquent via le passage de messages.
- Les microservices et les architectures sans serveur sont des descendants spirituels d'une architecture orientée services. Ils conservent bon nombre des principes et idées susmentionnés tout en essayant de résoudre la complexité des architectures orientées services à l'ancienne.

# Comment en est-on arrivé là ?

## Les architectures orientées service et les microservices



## **Section 2 :**

# **Qu'est ce que le calcul sans serveur ?**



# Qu'est ce que le calcul sans serveur ?

## Définition

### Définition

Les architectures sans serveur se réfèrent à des applications qui dépendent de manière significative de services tiers ou sur un code personnalisé exécuté dans des conteneurs éphémères.

### Attention !

Le terme «Serverless» est source de confusion car, dans de telles applications, il existe à la fois du matériel et des processus serveur. La différence avec les approches basiques est que l'entreprise créant et prenant en charge une application dite Serverless ne s'occupe pas du matériel ou des processus, ils externalisent cela à un fournisseur (AWS, Google, ...) et ainsi se concentrent uniquement sur la partie fonctionnelle de l'application.

# Qu'est ce que le calcul sans serveur ?

## Définition

- L'informatique sans serveur est une technologie, également connue sous le nom de fonction en tant que service (FaaS), qui offre au fournisseur de cloud une gestion complète du conteneur sur lequel les fonctions s'exécutent si nécessaire pour répondre aux demandes.
- Ce faisant, ces architectures éliminent le besoin de systèmes fonctionnant en continu et servent de calculs pilotés par les événements. La faisabilité de créer des applications évolutives au sein de cette architecture est énorme.
- Imaginez avoir la possibilité d'écrire simplement du code, de le télécharger et de l'exécuter, sans avoir à vous soucier de l'infrastructure sous-jacente, de la configuration ou de la maintenance de l'environnement... Les possibilités sont infinies et la vitesse de développement augmente rapidement.
- En utilisant une architecture sans serveur, vous pouvez déployer des applications entièrement fonctionnelles et évolutives dans la moitié du temps qu'il vous faut pour les construire à partir de zéro.



# Qu'est ce que le calcul sans serveur ?

## Sans serveur en tant que calcul piloté par les événements

- Le calcul piloté par les événements est un modèle d'architecture qui met l'accent sur l'action en réponse ou basée sur la réception d'événements.
- Ce modèle favorise les services à couplage faible et garantit qu'une fonction ne s'exécute que lorsqu'elle est déclenchée.
- Il encourage également les développeurs à réfléchir aux types d'événements et de réponses dont une fonction a besoin pour gérer ces événements avant de programmer la fonction.
- Dans cette architecture événementielle, les fonctions sont des consommateurs d'événements car elles sont censées prendre vie lorsqu'un événement se produit et sont responsables de leur traitement.
- Exemples : Requettes API, Objets placés et récupérés dans le stockage d'objets, Modifications apportées aux éléments de la base de données, Évènements planifiés, Commandes vocales (par exemple, Amazon Alexa), Bots (tels que AWS Lex et Azure LUIS, deux moteurs de traitement en langage naturel).

# Qu'est ce que le calcul sans serveur ?

## Fonction en tant que Service (FaaS)

Gérés par le fournisseur   
Gérés par l'organisation / le client 

Application	Application	Application	Application	Application
Données	Données	Données	Données	Données
Middleware	Middleware	Middleware	Middleware	Middleware
SE	SE	SE	SE	SE
Virtualisation	Virtualisation	Virtualisation	Virtualisation	Virtualisation
Serveurs	Serveurs	Serveurs	Serveurs	Serveurs
Stockage	Stockage	Stockage	Stockage	Stockage
Réseaux	Réseaux	Réseaux	Réseaux	Réseaux

On premise / cloud  
privé

IaaS



PaaS

SaaS

FaaS

# Qu'est ce que le calcul sans serveur ?

## Fonction en tant que Service (FaaS)

Gérés par le fournisseur   
Gérés par l'organisation / le client 

Application	Application	Application	Application	Application
Données	Données	Données	Données	Données
Middleware	Middleware	Middleware	Middleware	Middleware
SE	SE	SE	SE	SE
Virtualisation	Virtualisation	Virtualisation	Virtualisation	Virtualisation
Serveurs	Serveurs	Serveurs	Serveurs	Serveurs
Stockage	Stockage	Stockage	Stockage	Stockage
Réseaux	Réseaux	Réseaux	Réseaux	Réseaux

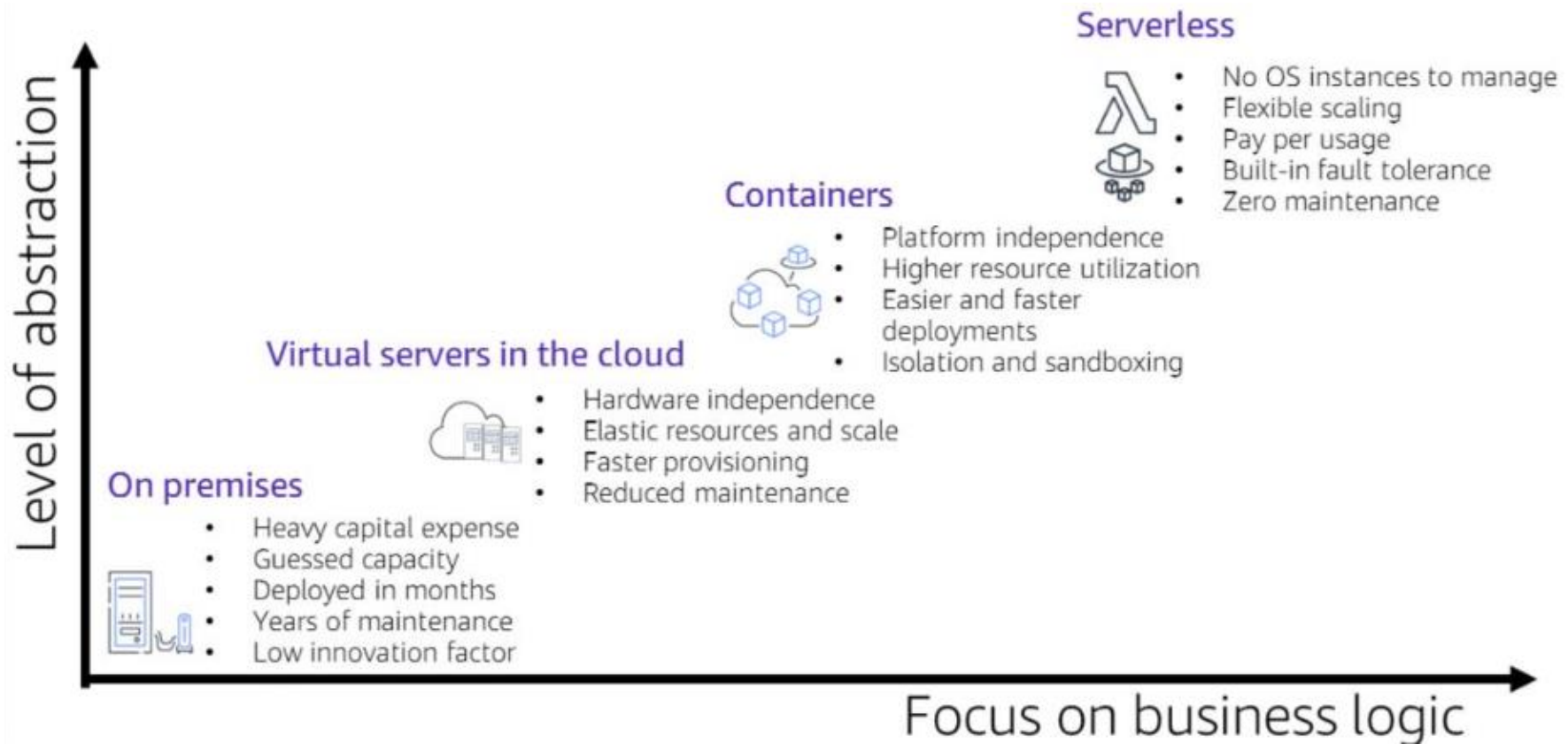
On premise / cloud  
privé

IaaS

PaaS

SaaS

FaaS



# **Section 3 :**

## **Comment le calcul sans serveur fonctionne ?**

# Comment le calcul sans serveur fonctionne ?

## Fonctionnement du calcul sans serveur

- Disons que nous avons un site Web où nous pouvons rechercher et acheter des manuels. Dans une architecture traditionnelle, vous pourriez avoir un client, un serveur à charge équilibrée et une base de données pour les manuels.
- Dans une architecture sans serveur, plusieurs choses peuvent changer, notamment le serveur et la base de données. Un exemple de ce changement serait la création d'une API provisionnée dans le cloud et le mappage de demandes de méthode spécifiques à différentes fonctions.
- Au lieu d'avoir un serveur, notre application dispose désormais de fonctions pour chaque élément de fonctionnalité et de serveurs provisionnés dans le cloud qui sont créés en fonction de la demande.
- Nous pourrions avoir une fonction pour rechercher un livre, et aussi une fonction pour acheter un livre. Nous pourrions également choisir de diviser notre base de données en deux bases de données distinctes qui correspondent aux deux fonctions.
- un événement déclenchera l'exécution du code d'application, puis le fournisseur de cloud alloue dynamiquement les ressources pour ce code, et l'utilisateur cesse de payer lorsque le code a fini de s'exécuter.



# Comment le calcul sans serveur fonctionne ?

## Méthode de réalisation du calcul sans serveur

- Il existe deux méthodes principales de calcul sans serveur.
- Le premier est via Backend-as-a-Service (BaaS), où une variété de services et d'applications tiers composent votre application.
- La seconde est via Function-as-a-Service (FaaS), où les développeurs écrivent toujours une logique côté serveur personnalisée, mais elle est exécutée dans des conteneurs entièrement gérés par un fournisseur de cloud.
- On peut également noter que vous puissiez créer une application entièrement sans serveur grâce à ces méthodes, ou faire un compromis avec une application de composants de microservices partiellement sans serveur et partiellement traditionnels.
- Le backend en tant que service (BaaS) est une méthode de calcul sans serveur qui repose largement sur des applications et des services tiers.
- Function-as-a-Service (FaaS) inclut un plus grand degré de contrôle qu'un BaaS, car la logique côté serveur est toujours écrite par les développeurs. Cependant, une fois écrit, il est déployé dans des conteneurs gérés par un fournisseur de cloud, ce qui constitue le principal avantage du calcul sans serveur.

## **Section 3 : Comment c'est différent ?**

# Comment c'est différent ?

## Développement

- Le processus de développement pour les applications sans serveur change légèrement de la façon dont on développerait un système sur site. Les aspects de l'environnement de développement, y compris les IDE, le contrôle des sources, la gestion des versions et les options de déploiement, peuvent tous être définis par le développeur sur site ou avec le fournisseur de cloud.
- Une méthode préférée de développement continu consiste à écrire des fonctions sans serveur à l'aide d'un IDE, comme Visual Studio, Eclipse et IntelliJ, et à le déployer en petits morceaux sur le fournisseur de cloud à l'aide de l'interface de ligne de commande du fournisseur de cloud.
- Si les fonctions sont suffisamment petites, elles peuvent être développées dans le portail du fournisseur de cloud. Cependant, la plupart ont une limite sur la taille de la fonction avant d'exiger un téléchargement zip du projet.
- L'interface de ligne de commande (CLI) est un puissant outil de développement car elle rend les fonctions sans serveur et leurs services nécessaires facilement déployables et vous permet de continuer à utiliser les outils de développement que vous souhaitez utiliser pour écrire et produire votre code.

# Comment c'est différent ?

## Les processus indépendants

- Une autre façon de voir les fonctions sans serveur est comme des microservices sans serveur.
- Chaque fonction sert son propre objectif et complète un processus indépendamment des autres fonctions.
- Le calcul sans serveur est sans état (stateless) et basée sur les événements, c'est ainsi que les fonctions doivent également être développées.
- Par exemple, dans une architecture traditionnelle avec des opérations de base API CRUD (GET, POST, PUT, DELETE), vous pouvez avoir des modèles basés sur des objets avec ces méthodes définies sur chaque objet. L'idée de maintenir la modularité s'applique toujours au niveau sans serveur. Chaque fonction peut représenter une méthode API et effectuer un processus.

# **Section 4 :**

## **Exemple d'utilisation (AWS Lambda)**

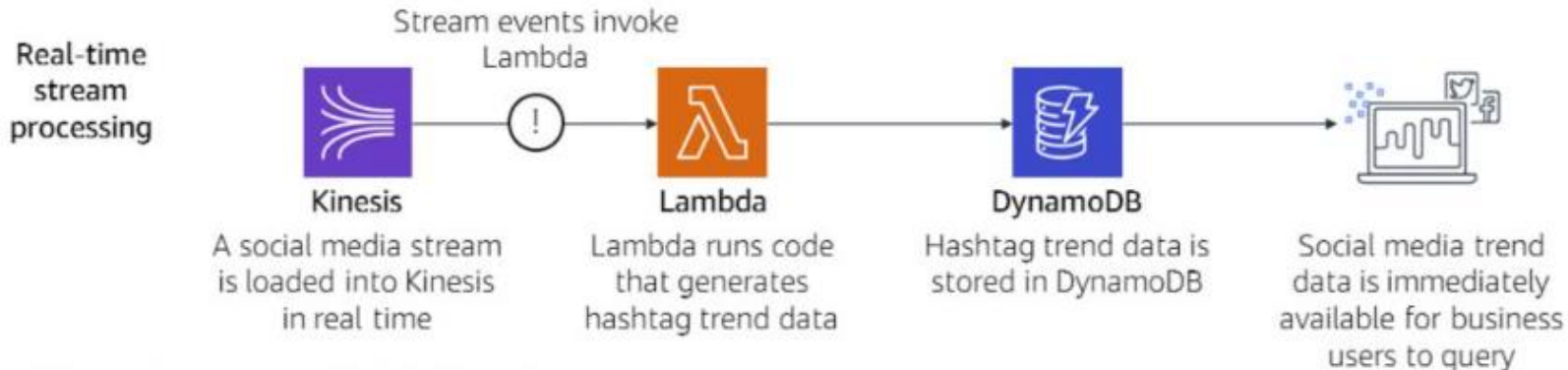
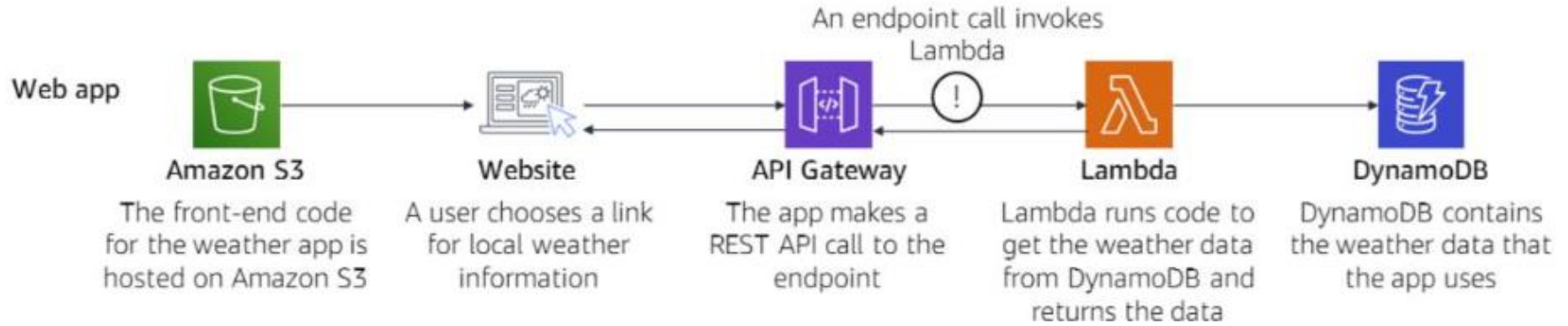
# Exemples d'utilisation (AWS Lambda)

Vous pouvez utiliser Lambda de différentes manières. Les applications Web et le traitement de flux en temps réel sont deux cas d'utilisation courants. D'autres cas d'utilisation courants sont les suivants :

- **Backends** : vous pouvez créer des backends sans serveur en utilisant Lambda pour gérer les requêtes Web, mobiles, Internet des objets (IoT) et d'API tierces. Vous pouvez créer des backends en utilisant Lambda et Amazon API Gateway pour authentifier et traiter les demandes d'API.
- **Traitement des données** : vous pouvez utiliser Lambda pour exécuter du code en réponse à des déclencheurs, tels que des modifications de données, des changements d'état du système ou des actions d'utilisateurs.
- **Chatbots**: Amazon Lex est un service permettant de créer des interfaces conversationnelles dans n'importe quelle application en utilisant la voix et le texte. Amazon Lex utilise les mêmes technologies d'apprentissage en profondeur qui alimentent Amazon Alexa et s'intègre à Lambda.
- **Amazon Alexa**: Vous pouvez utiliser Lambda pour créer le backend pour les compétences Alexa personnalisées
- **Automatisation informatique** : Vous pouvez utiliser Lambda pour automatiser les processus répétitifs en les invoquant avec des événements ou en les exécutant selon un calendrier fixe. Vous pouvez automatiquement mettre à jour le micrologiciel des appareils matériels, démarrer et arrêter les instances Amazon Elastic Compute Cloud (Amazon EC2) ou planifier des mises à jour des groupes de sécurité. De plus, vous pouvez automatiser votre pipeline de test et de déploiement.

**Tout ce que vous voulez faire avec les serveurs, vous pouvez probablement le faire avec Lambda. Utilisez Lambda et sans serveur pour réinventer votre approche des applications nouvelles ou mises à jour.**

## Exemples d'utilisation (AWS Lambda)



## **Section 4 : Avantages et limites**



# Avantages et limites

## Avantages

- **Développement et déploiement rapides** : Étant donné que toute l'infrastructure, la maintenance et la mise à l'échelle automatique sont gérées par le fournisseur de cloud, le temps de développement est beaucoup plus rapide et le déploiement plus facile qu'auparavant. Le développeur n'est responsable que de l'application elle-même, supprimant la nécessité de planifier le temps à consacrer à la configuration du serveur.
- **Facilité d'utilisation** : La plus grande partie de cette simplicité est due à des services, fournis par des fournisseurs de cloud, qui facilitent la mise en œuvre de solutions complètes. Les événements nécessaires à l'exécution de votre fonction sont facilement créés et provisionnés dans l'environnement cloud et nécessitent peu de maintenance.
- **Moindre coût** : Pour les solutions sans serveur, vous êtes facturé par exécution plutôt que par l'existence de l'ensemble des applications. Cela signifie que vous payez exactement ce que vous utilisez. De plus, comme les serveurs de l'application sont gérés et mis à l'échelle automatiquement par un fournisseur de cloud, ils ont également un prix moins cher que ce que vous paieriez en interne.
- **Évolutivité améliorée** : Avec les solutions sans serveur, l'évolutivité est automatiquement intégrée car les serveurs sont gérés par des fournisseurs tiers. Cela signifie que le temps, l'argent et l'analyse généralement consacrés à la configuration de la mise à l'échelle et de l'équilibrage automatiques sont supprimés. En plus de l'évolutivité, la disponibilité est également augmentée car les fournisseurs de cloud maintiennent la capacité de calcul dans toutes les zones et régions de disponibilité. Cela rend votre application sans serveur sécurisée et disponible car elle protège le code des défaillances régionales.

# Avantages et limites

## Limites

- Contrôle des infrastructures : Une limite potentielle pour une architecture sans serveur est la nécessité de contrôler l'infrastructure. Bien que les fournisseurs de cloud conservent le contrôle et le provisionnement de l'infrastructure et du système d'exploitation, cela ne signifie pas que les développeurs perdent la capacité de déterminer les éléments de l'infrastructure. Dans le portail de fonctions de chaque fournisseur de cloud, les utilisateurs peuvent choisir le temps d'exécution, la mémoire, les autorisations et le délai d'expiration. De cette façon, le développeur a toujours le contrôle sans la maintenance.
- Application serveur longue durée : L'un des avantages des architectures sans serveur est qu'elles sont conçues pour être des fonctions rapides, évolutives et événementielles. Par conséquent, les opérations par lots de longue durée ne conviennent pas bien à cette architecture. La plupart des fournisseurs de cloud ont un délai d'expiration de cinq minutes, donc tout processus qui prend plus de temps que ce temps alloué est interrompu. L'idée est de s'éloigner du traitement par lots pour passer à une fonctionnalité en temps réel, rapide et réactive.

# Avantages et limites

## Limites

- Verrouillage fournisseur : L'une des plus grandes craintes de passer à la technologie sans serveur est celle du verrouillage des fournisseurs. C'est une crainte commune à tout passage à la technologie cloud. Les entreprises craignent qu'en s'engageant à utiliser Lambda, elles s'engagent dans AWS et ne pourront pas passer à un autre fournisseur de cloud ou ne pourront pas se permettre une autre transition vers un fournisseur de cloud.
- Démarrage à froid : Le souci d'un «démarrage à froid» est qu'une fonction prend un peu plus de temps pour répondre à un événement après une période d'inactivité. Cela a tendance à se produire, mais il existe des moyens de contourner le démarrage à froid si vous avez besoin d'une fonction immédiatement réactive. Si vous savez que votre fonction ne sera déclenchée que périodiquement, une approche pour surmonter le démarrage à froid consiste à établir un planificateur qui appelle votre fonction pour la réveiller de temps en temps.
- Infrastructure partagée : Cela peut être une préoccupation d'un point de vue commercial, car les fonctions sans serveur peuvent fonctionner côte à côte indépendamment de la propriété de l'entreprise.

# Conclusion