



# AWS Serverless

Par Lahda Biassou Alphonsine



# Lahda Biassou Alphonsine

## Ingénieure cloud et formatrice





# Plan

- **Vue globale de AWS Serverless**
- **Amazon EKS**
- **Amazon ECS**
- **Amazon ECR**
- **AWS Fargate**
- **Amazon X-Ray**
- **AWS Lambda**
- **Amazon API Gateway**
- **AWS AppSync**
- **AWS SAM**
- **Amazon Cloudfront**
- **ElastiCache**
- **Amazon Dax**





# Vue globale de AWS Serverless -définition

Une solution qui ne considère pas la gestion des serveurs

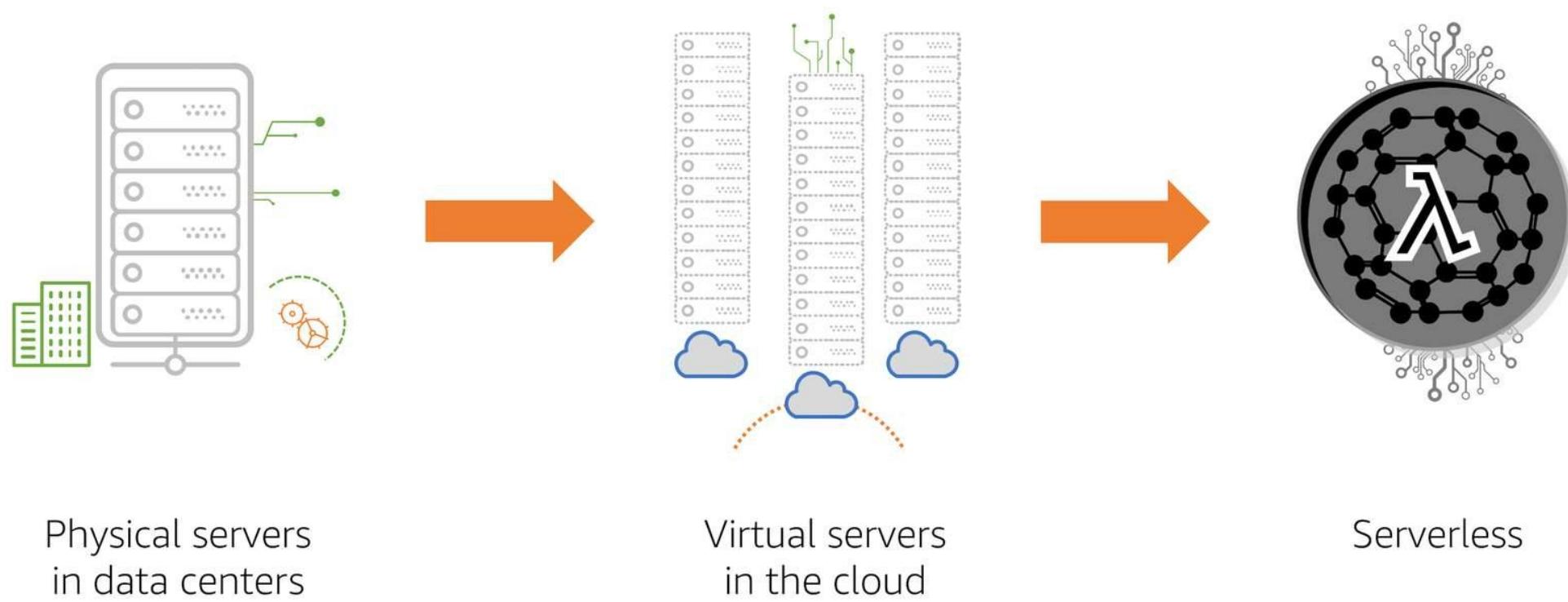
Flexible et scalable

Haute disponibilité automatisée

Vous ne payez jamais pour une capacité inutilisée.

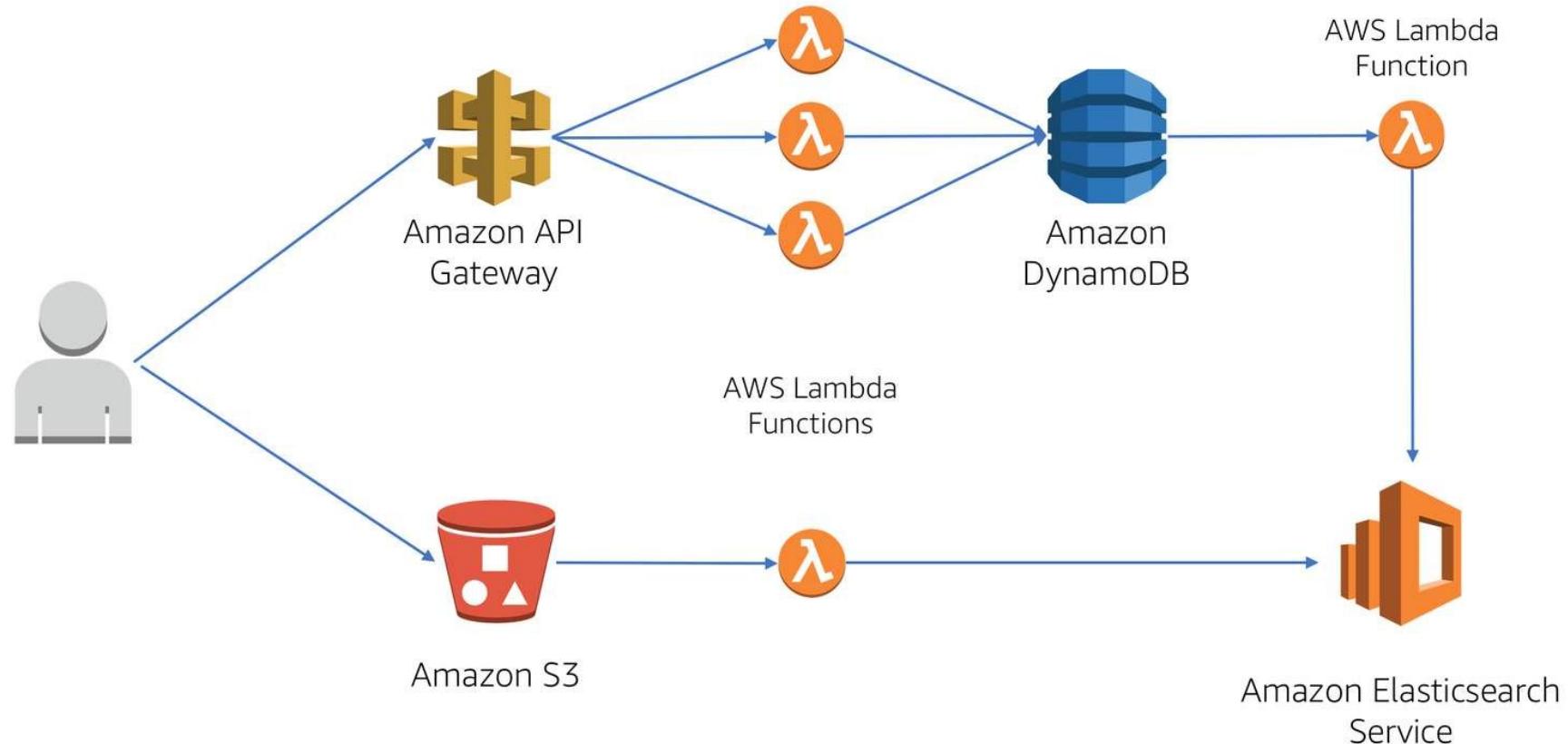


# AWS Serverless -amplification de l'avancé du cloud





# AWS Serverless -exemple d'architecture





# Avantages de Serverless

## Facile à utiliser

Vous vous occupez de votre code et tout le reste le serverless s'en occupe

## Basé sur les événements

Les ressources informatiques sans serveur peuvent être achetées en ligne en réponse à un événement qui se produit.

## Modèle de tarification

“Pay as you go” est le modèle de tarification. Vous ne payez que pour ce que pour les ressources provisionnées ou le temps d'exécution.



# Plan

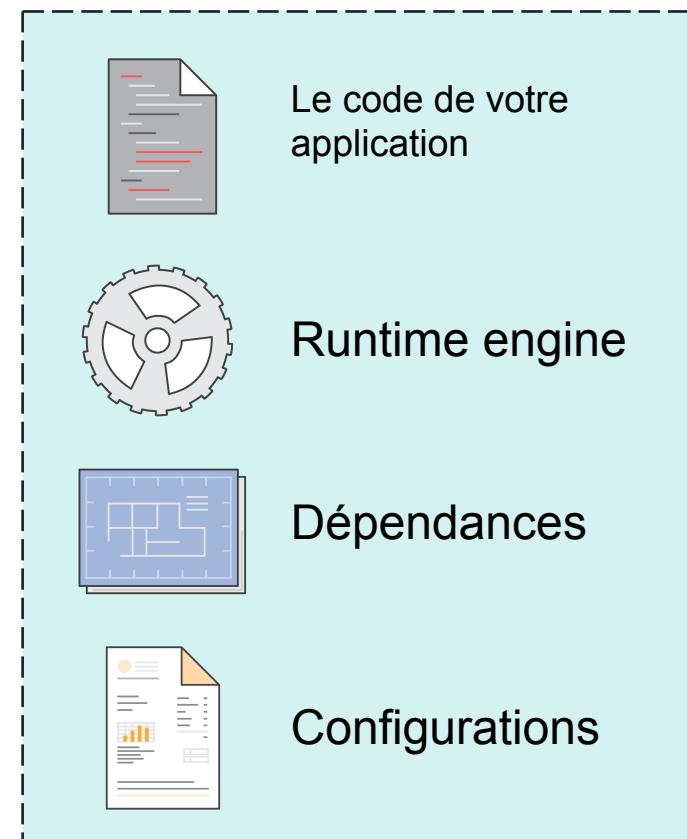
- Vue globale de AWS Serverless
- [Amazon EKS](#)
- [Amazon ECS](#)
- [Amazon ECR](#)
- [AWS Fargate](#)
- [Amazon X-Ray](#)
- [AWS Lambda](#)
- [Amazon API Gateway](#)
- [AWS AppSync](#)
- [AWS SAM](#)
- [Amazon Cloudfront](#)
- [ElastiCache](#)
- [Amazon Dax](#)





# C'est quoi les conteneurs?

## Votre conteneur



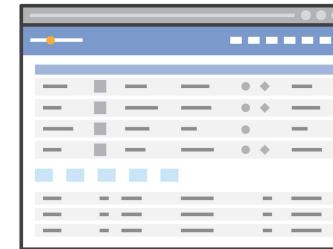


## Quel problème les conteneurs résout?

Faire en sorte que les logiciels fonctionnent de manière fiable dans différents environnements de travail



Poste de travail du développeur



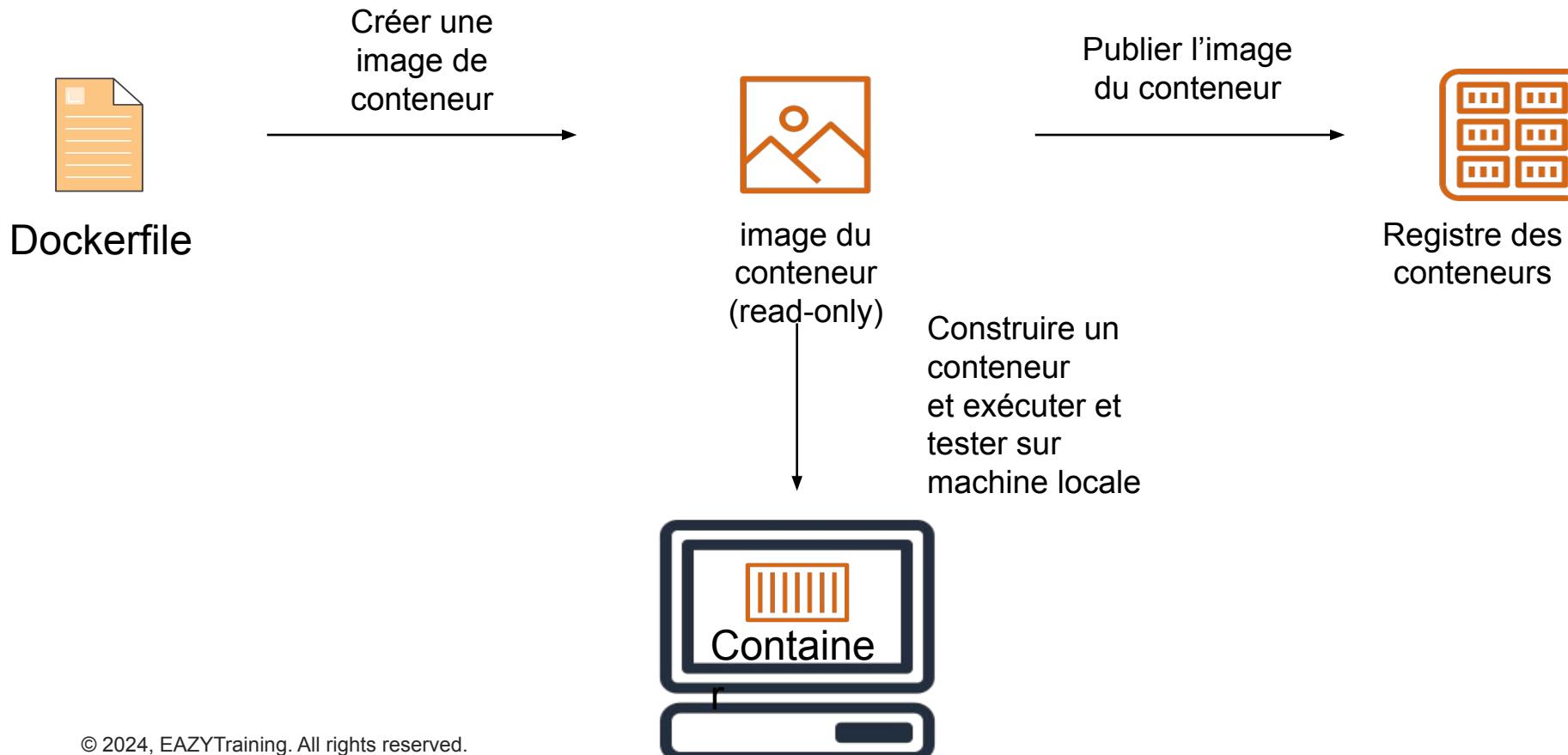
Environnement de production



Environnement de test



# Terminologies des conteneurs





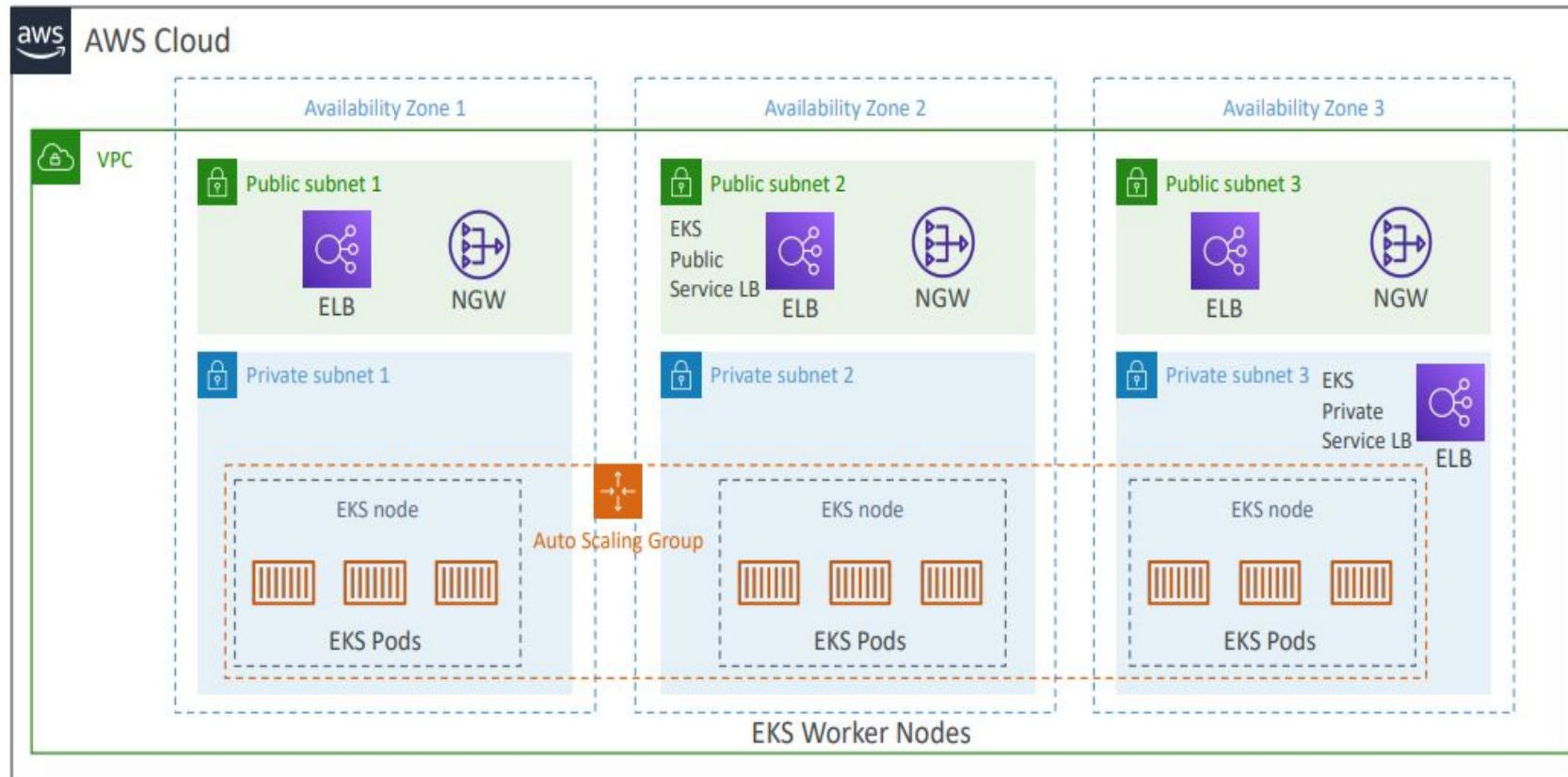
## Amazon EKS - overview



- Amazon EKS = Amazon Elastic Kubernetes Service
- Il s'agit d'une manière de lancer des clusters Kubernetes gérés sur AWS.
- Kubernetes est un système open-source pour le déploiement automatique, la mise à l'échelle et la gestion des applications conteneurisées (généralement Docker).
- Il s'agit d'une alternative à ECS, avec un objectif similaire mais une API différente.
- EKS prend en charge EC2 si vous souhaitez déployer des nœuds de travail ou Fargate pour déployer des conteneurs sans serveur.
- Cas d'utilisation : si votre entreprise utilise déjà Kubernetes sur site ou dans un autre cloud, et souhaite migrer vers AWS en utilisant Kubernetes.
- Kubernetes est agnostique (peut être utilisé dans n'importe quel cloud - Azure, GCP...)
- Pour plusieurs régions, déployer un cluster EKS par région
- Collecter des logs et des métriques à l'aide de CloudWatch Container Insights



# Amazon EKS - overview





# Amazon EKS - Types de noeuds

- **Groupes de noeuds gérés**
  - Crée et gère des noeuds (instances EC2) pour vous.
  - Les noeuds font partie d'un ASG géré par EKS
  - Supporte les instances à la demande ou ponctuelles
- **Noeuds autogérés**
  - Noeuds créés par vous, enregistrés dans le cluster EKS et gérés par un ASG.
  - Vous pouvez utiliser une AMI préconstruite - Amazon EKS Optimized AMI
  - Prend en charge les instances à la demande ou ponctuelles
- **Fargate AWS**
  - Aucune maintenance n'est requise ; aucun noeud n'est géré



# Amazon EKS Anywhere





# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- **Amazon ECS**
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**

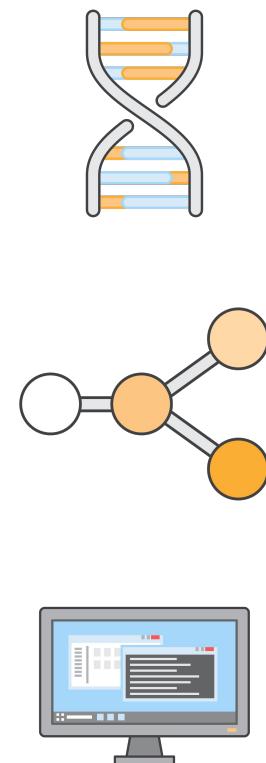




## Amazon Elastic Container Service (ECS)



Amazon  
Elastic  
Container  
Service  
(Amazon ECS)



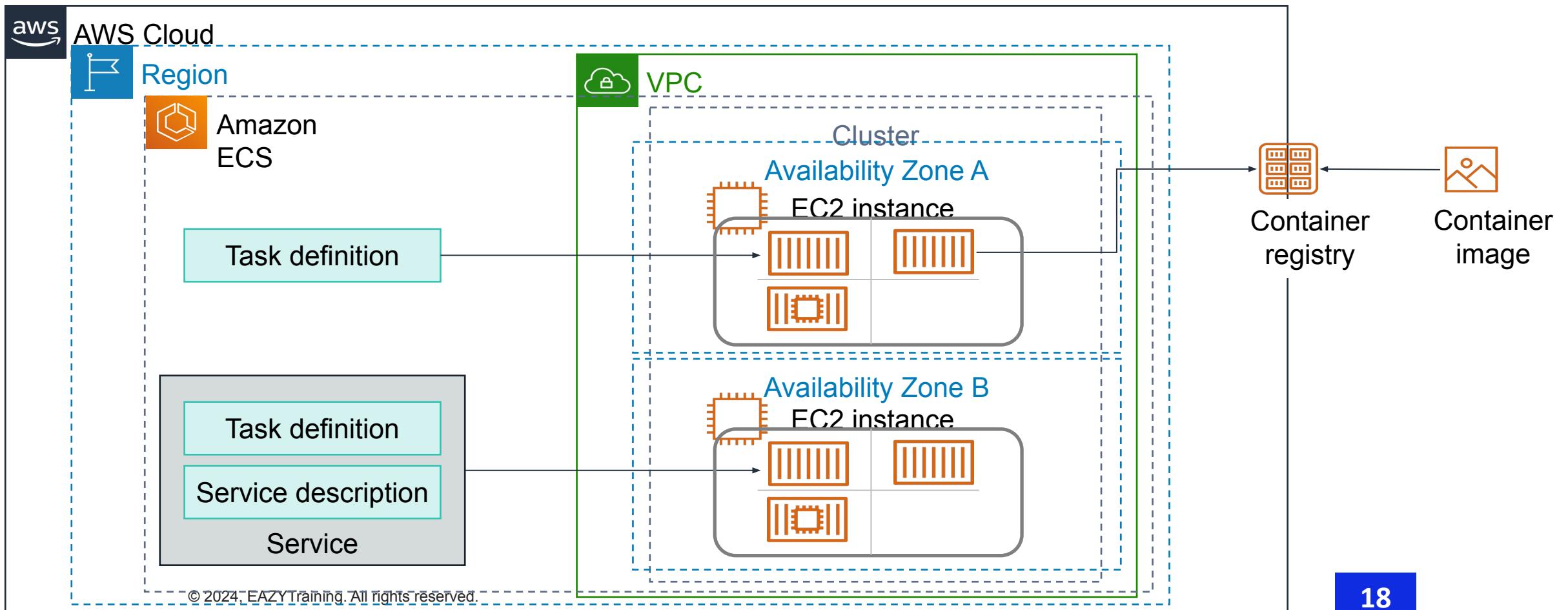
Orchestrer l'exécution des conteneurs

Maintient et fait évoluer la flotte d'instances qui exécutent vos conteneurs.

Supprime la complexité de la mise en place de l'infrastructure



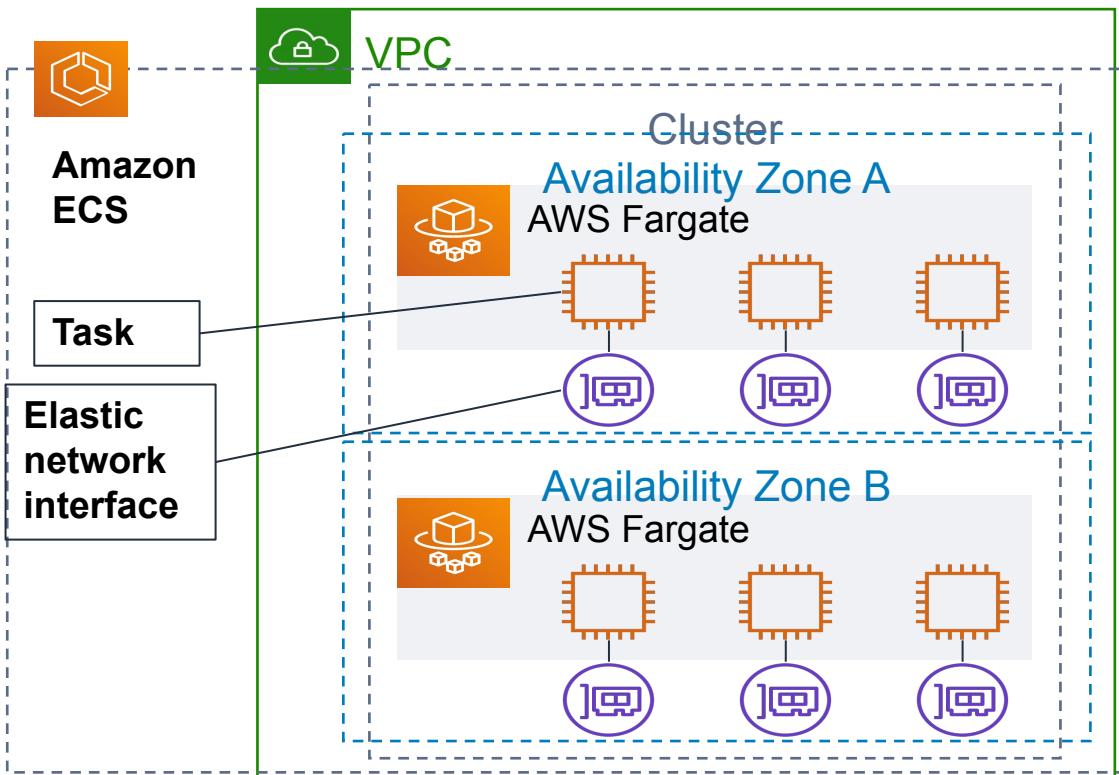
# Amazon ECS -orchestrer les conteneurs



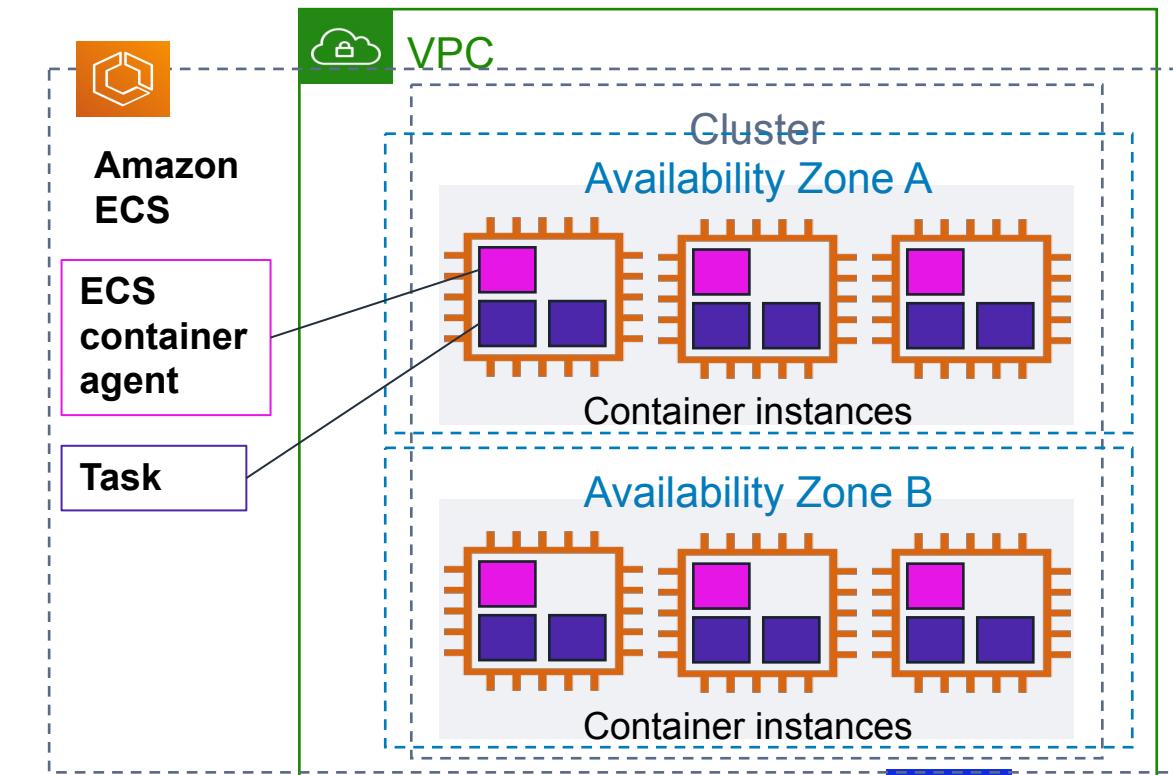


# Amazon ECS -types de lancement

## Fargate launch type

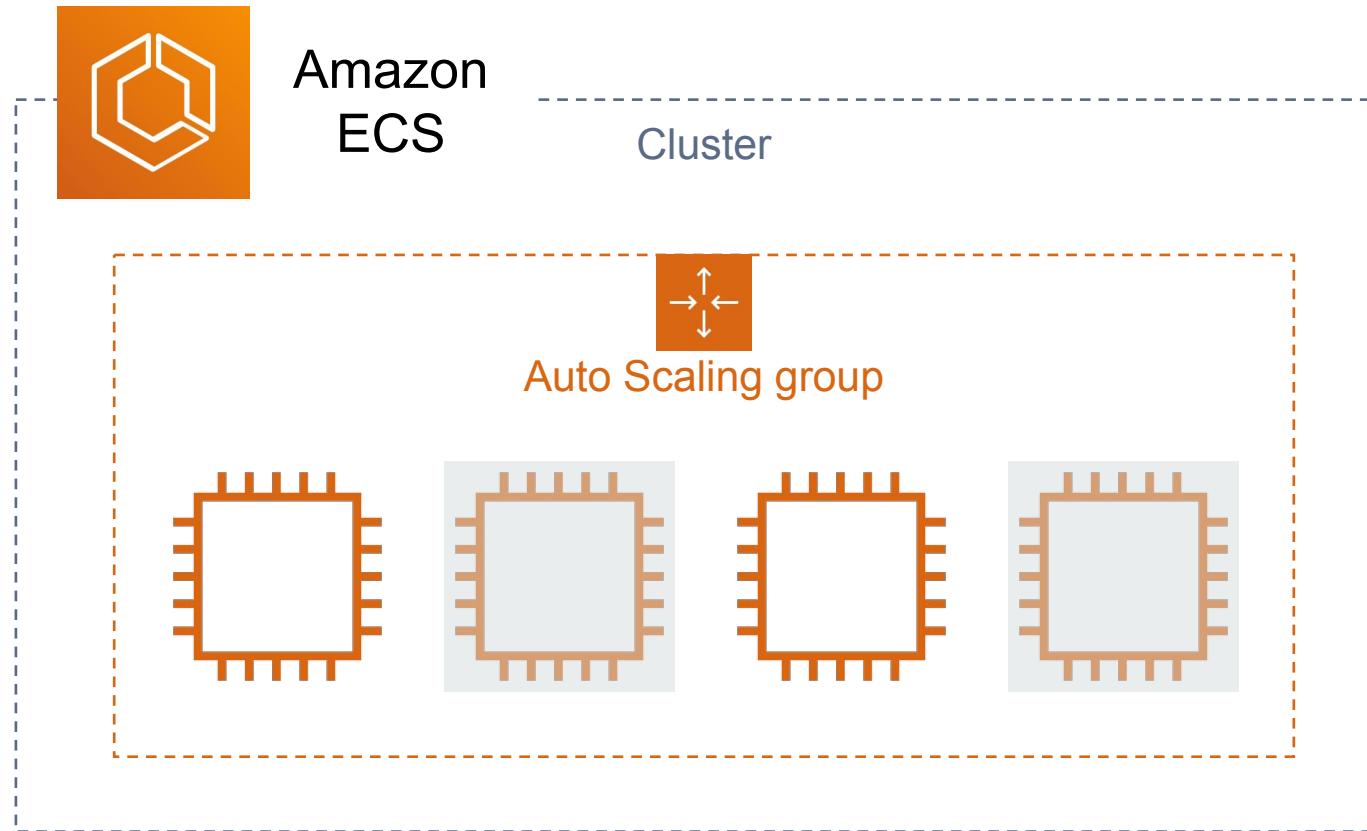


## EC2 launch type



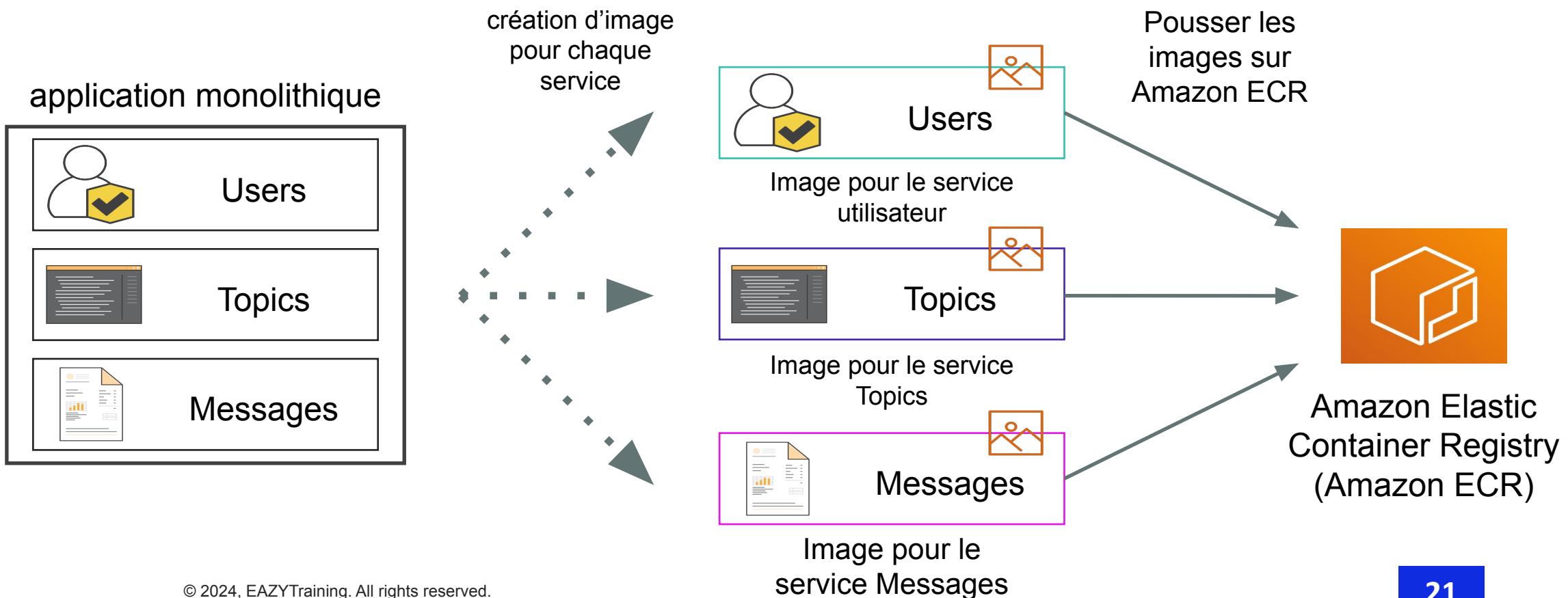


# Amazon ECS cluster auto scaling





## Décomposition d'une application monolithe - étape 1 création des images de conteneurs





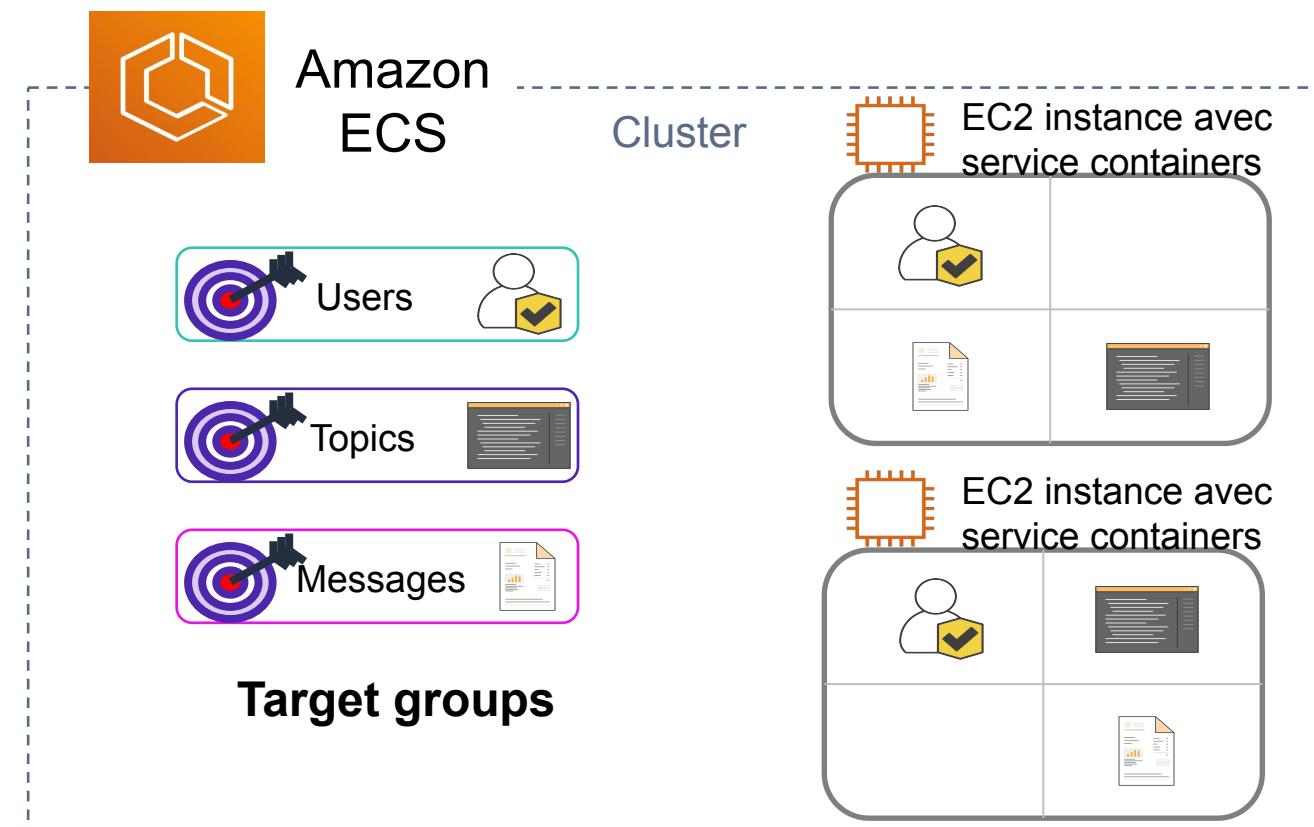
## Décomposition d'une application monolithe - etape 2 création des groupes des task definition et les target groups

### Service Task Definition

- Launch type = [EC2 or Fargate]
- Name = [service-name]
- Image = [service ECR repo URL]:version
- CPU = [256]
- Memory = [256]
- Container port = [3000]
- Host port = [0]

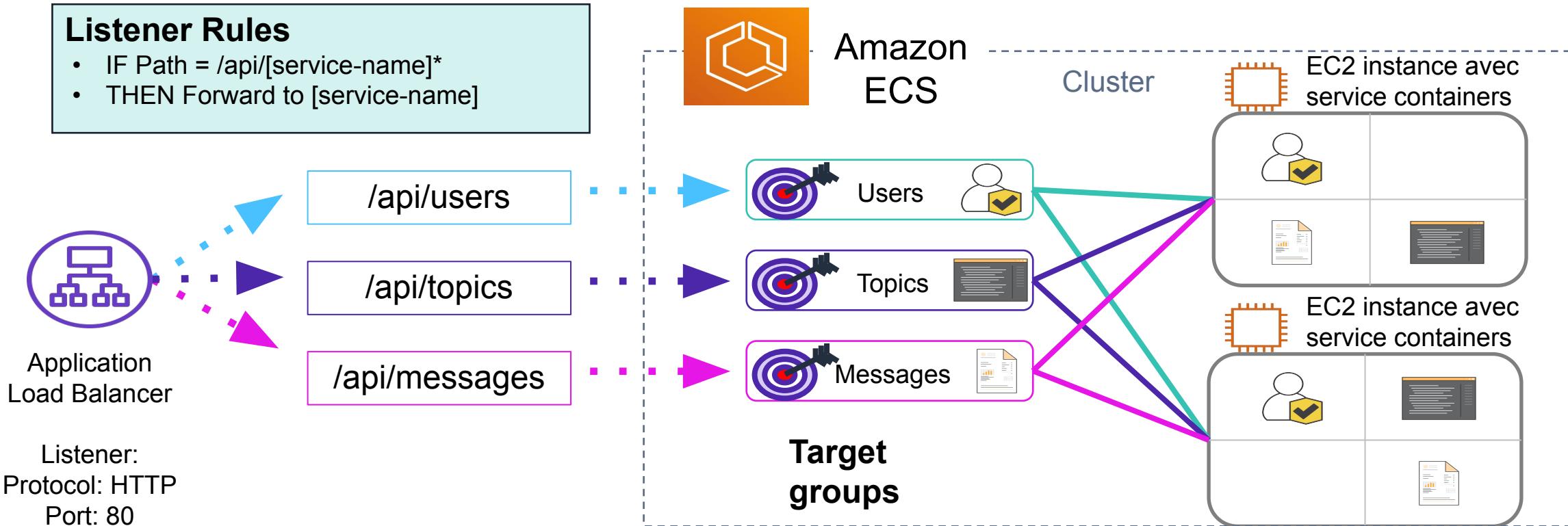
### Service Target Group

- Name = [service-name]
- Protocol = [HTTP]
- Port = [80]
- VPC = [vpc-name]



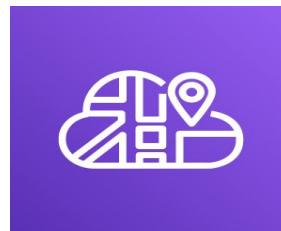


## Décomposition d'une application monolithe - etape 3 connecter les load balancers aux services





## Outils pour construire une architectures micro service hautement disponible



AWS  
Cloud Map

- Il s'agit d'un service de découverte entièrement géré pour les ressources en nuage
- Peut être utilisé pour définir des noms personnalisés pour les ressources d'application
- Maintient à jour l'emplacement des ressources qui changent dynamiquement, ce qui augmente la disponibilité des applications.



AWS  
App Mesh

- Capture les métriques, les journaux et les traces de tous vos microservices.
- vous permet d'exporter ces données vers Amazon CloudWatch, AWS X-Ray et les outils communautaires et partenaires compatibles du réseau de partenaires AWS (APN)
- Permet de contrôler les flux de trafic entre les microservices afin de garantir la haute disponibilité des services.



# Plan

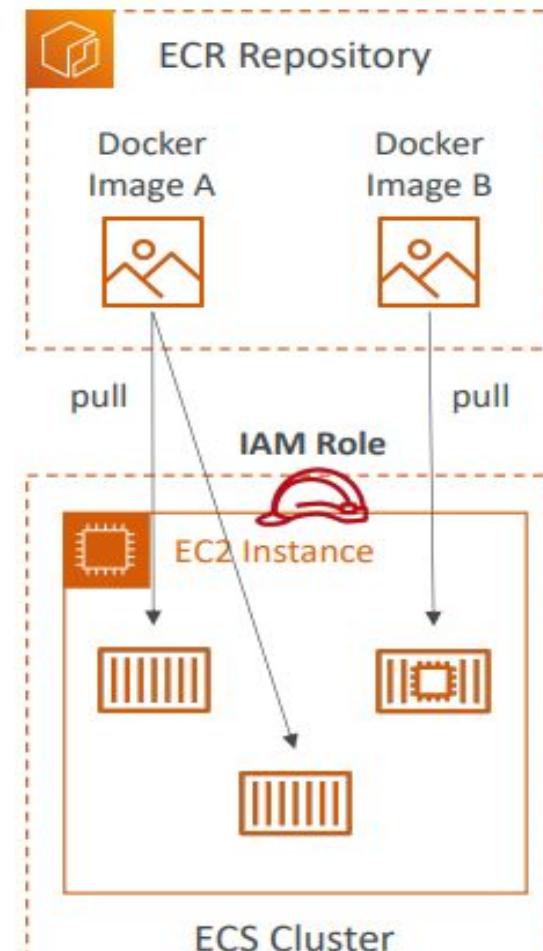
- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- [Amazon ECR](#)
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- [Amazon Dax](#)





# Amazon Elastic Container Registry (ECR)

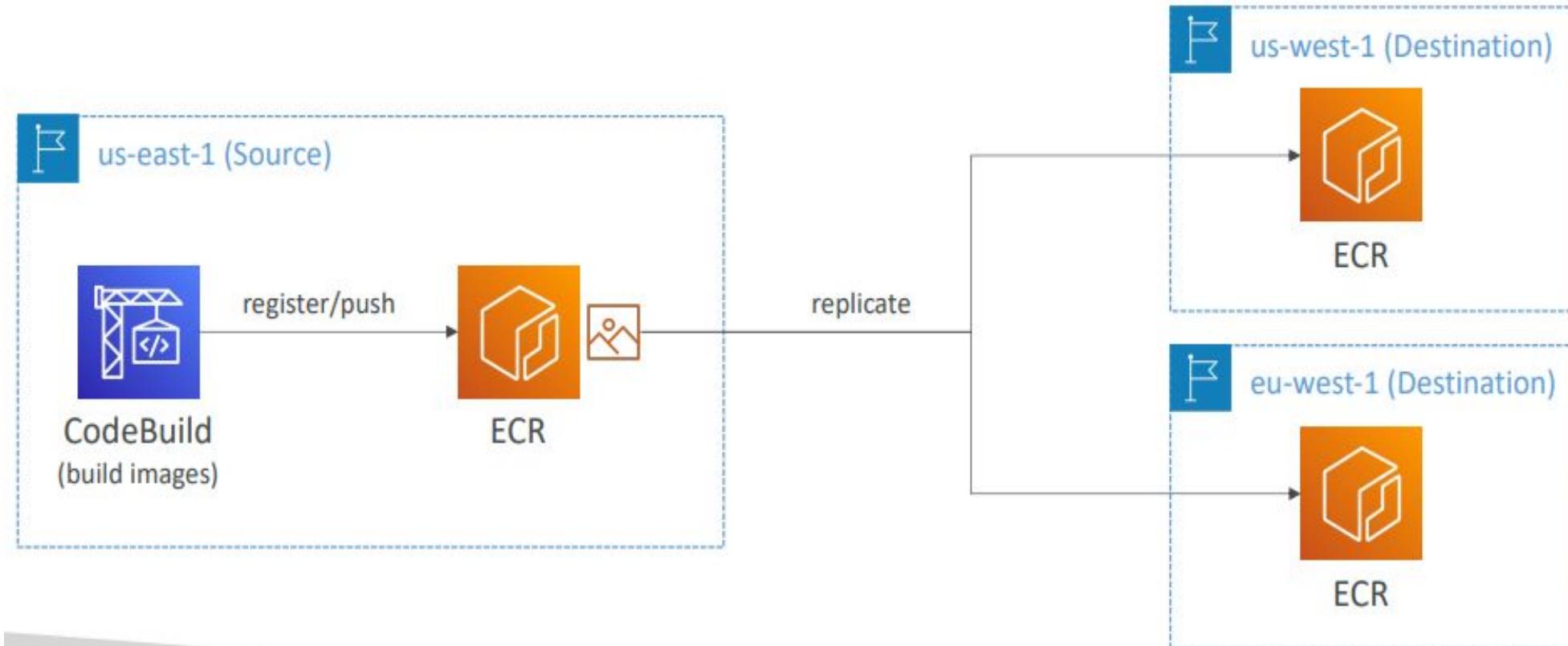
- Stocker et gérer des images Docker sur AWS
- Dépôt privé et public (Amazon ECR Galerie publique <https://gallery.ecr.aws>)
- Entièrement intégré à ECS
- L'accès est contrôlé par IAM (erreurs de permission => vérifier la politique)
- Prise en charge de l'analyse de la vulnérabilité des images, versions, balises d'images, cycle de vie des images, ...





# Amazon ECR - cross-Region Replication

- Le registre privé ECR prend en charge la réplication entre régions et entre comptes.





# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- **AWS Fargate**
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





# AWS Fargate



AWS  
Fargate

- Est un service de conteneurs entièrement géré  
29
- Fonctionne avec Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS)
- Fournit, gère et fait évoluer vos clusters de conteneurs
- Gère l'environnement d'exécution
- Fournit une mise à l'échelle automatique



# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- **Amazon X-Ray**
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





# Amazon X-Ray

## Définition



AWS X-Ray est un service qui permet l'analyse visuelle ou permet de tracer les applications microservices.

Il fournit des informations de bout en bout sur la demande, réponse et les appels à d'autres ressources AWS en passant par les composants sous-jacents de l'application de plusieurs microservices.

Il crée un graphe de service en utilisant les données de traçage des ressources AWS .

Le graphique montre les informations concernant les appels aux services front-end et back-end pour traiter les demandes et poursuivre la procédure.

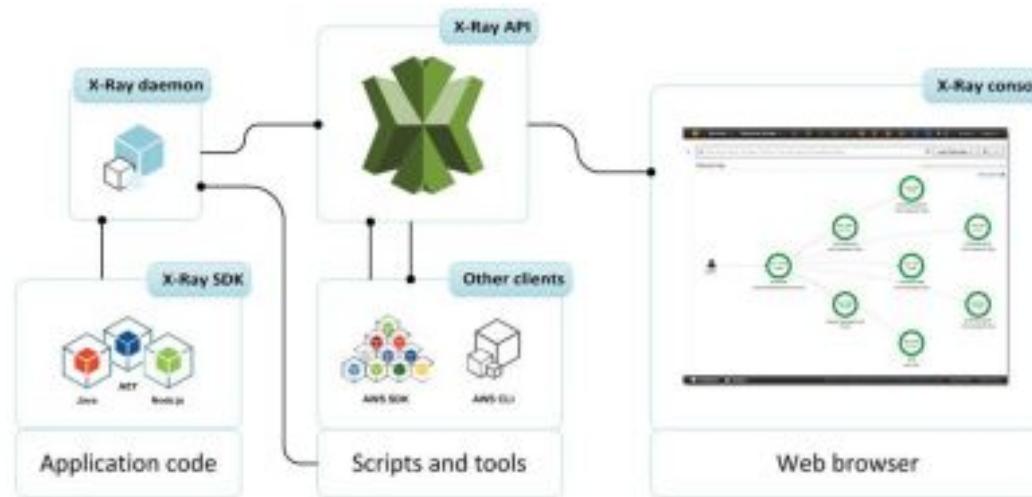
Le graphique aide à résoudre les problèmes et à améliorer les performances des applications.



# AWS X-Ray

→ X-Ray s'intègre avec les services suivants: EC2, Elastic Load Balancer, Elastic Beanstalk, Lambda, ECS et API Gateway

→ Les SDK X-Ray sont disponibles dans les langues suivantes : Go, Java, Node Js, Python Ruby et .Net





# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- **AWS Lambda**
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





# AWS Lambda

## Définition



AWS Lambda est un service AWS sans serveur qui permet aux utilisateurs d'exécuter du code sous forme de fonctions sans avoir à provisionner ou à gérer des serveurs.

Exécute les tâches administratives telles que la maintenance serveur, la journalisation, le provisionnement capacité, la mise à l'échelle automatique et la surveillance du code.

Possibilité de créer des applications sans serveur composées de fonctions Lambda déclenchées par des événements et peuvent être automatiquement déployées à l'aide d'AWS CodePipeline et AWS CodeBuild.



## AWS Lambda - caractéristiques



Apportez votre propre code



S'intégrer à d'autres services AWS



Ressources flexibles et résultats concurrentiels



Un modèle de permissions flexible



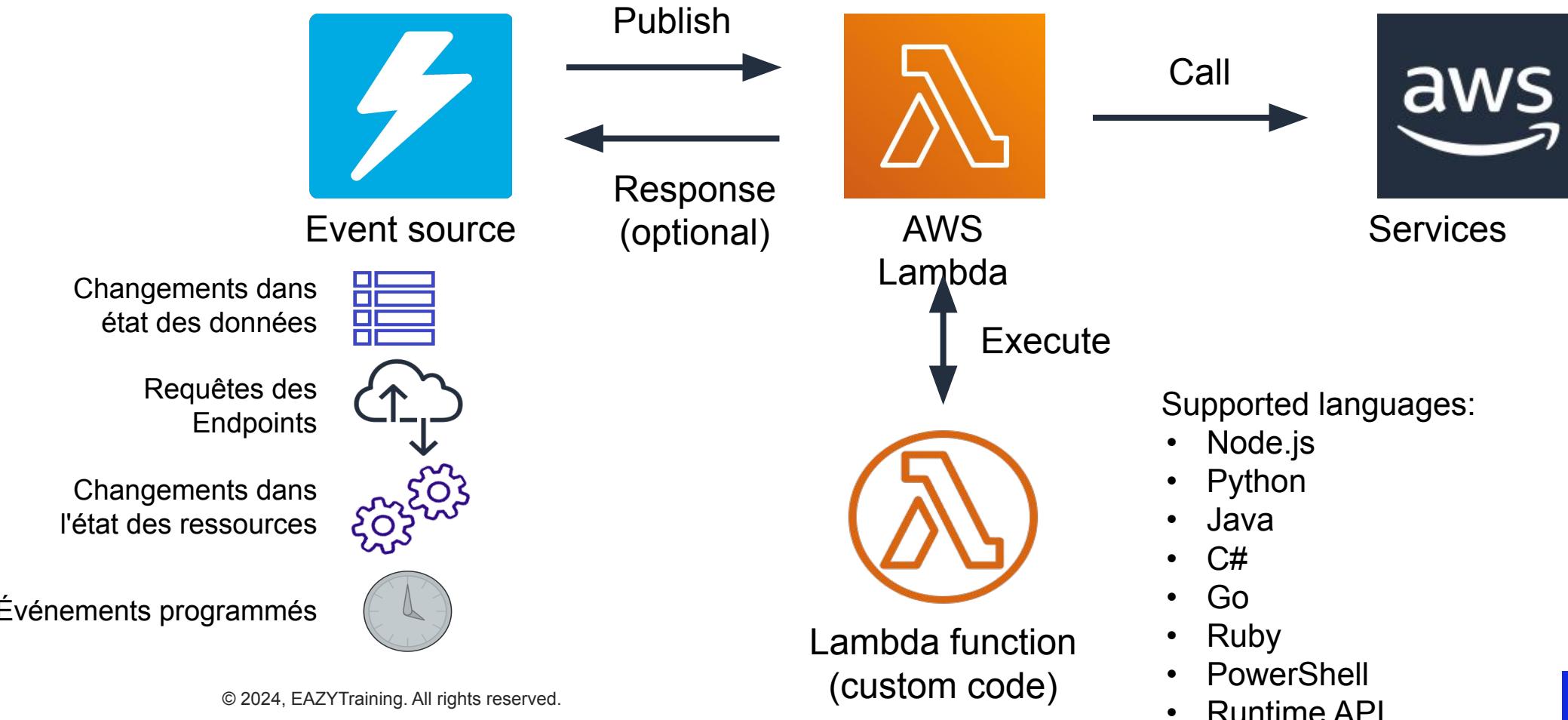
La disponibilité et la tolérance aux pannes sont intégrées



Payer en fonction de la valeur



# AWS Lambda - fonctionnement





# AWS Lambda - fonctions lambda

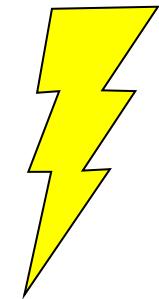


AWS  
Lambda

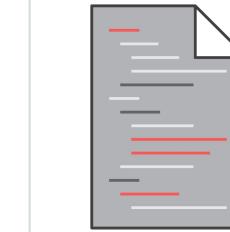
Lambda function



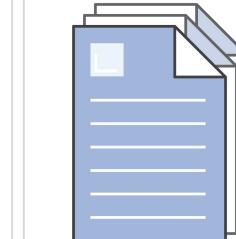
Access  
permissions



Triggering  
Events



Application  
code



Dependencies  
and libraries

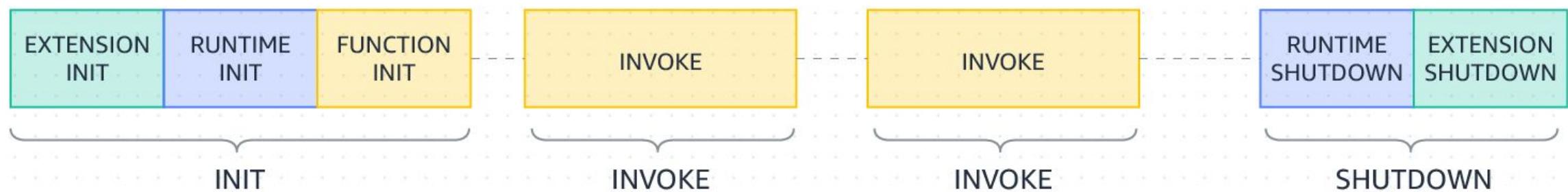


Configuration

Deployment package



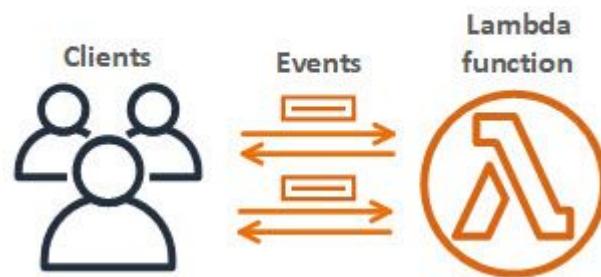
## Cycle de vie d'un environnement d'exécution Lambda



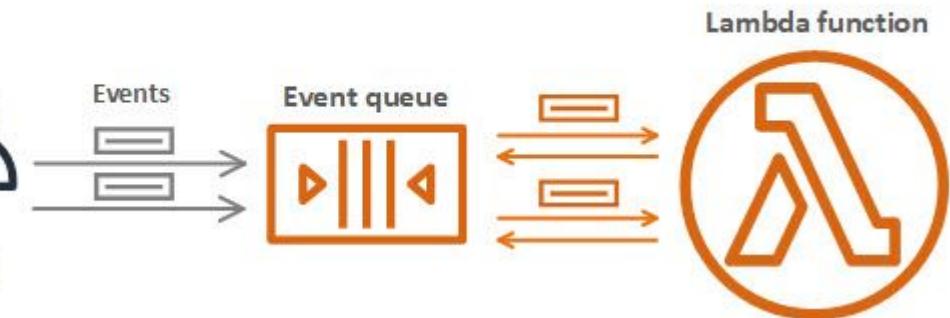


## AWS Lambda - modèles d'invocation de lambda

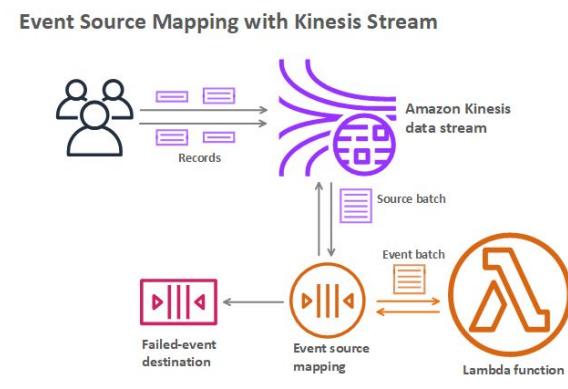
### L'invocation synchrone



### L'invocation asynchrone



### Mappage de source d'événements





# Anatomie de la fonction lambda

## Handler()

Fonction à exécuter lors de l'invocation

## Event object

Données envoyées lors de l'invocation de la fonction Lambda

## Context object

Méthodes disponibles pour interagir avec les informations d'exécution (ID de la demande, groupe de journaux, etc.)

```
import json
```

```
def lambda_handler(event, context):  
    # TODO implement  
    return {  
        'statusCode': 200,  
        'body': json.dumps('Hello World')  
    }
```



## Configuration et facturation de la fonction lambda

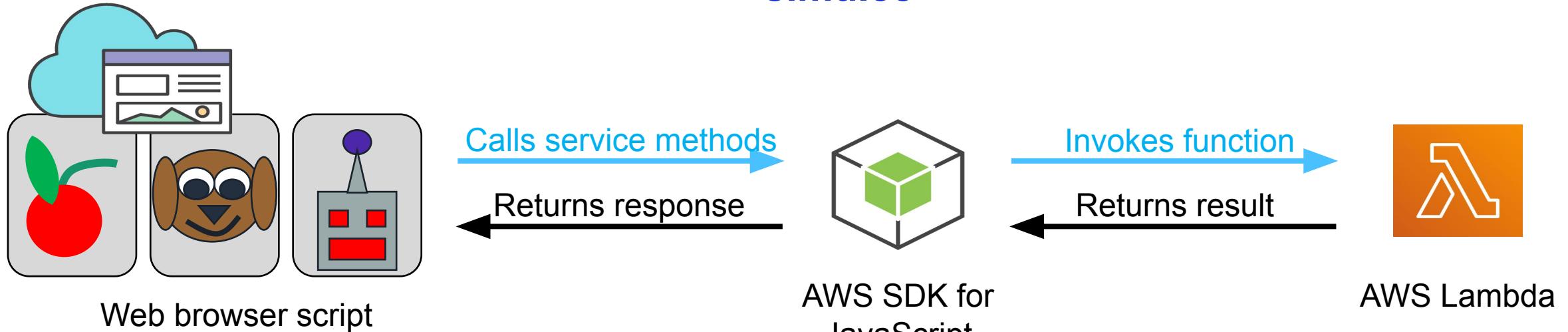
Mémoire - Le coût par 100 ms de la durée de la fonction augmente avec la mémoire.

Timeout - Vous contrôlez la durée maximale de votre fonction.

Tarification - Vous êtes facturé sur la base du nombre de demandes et de la durée.



## AWS Lambda - exemple sur le jeu par navigateur de machine à sous simulée



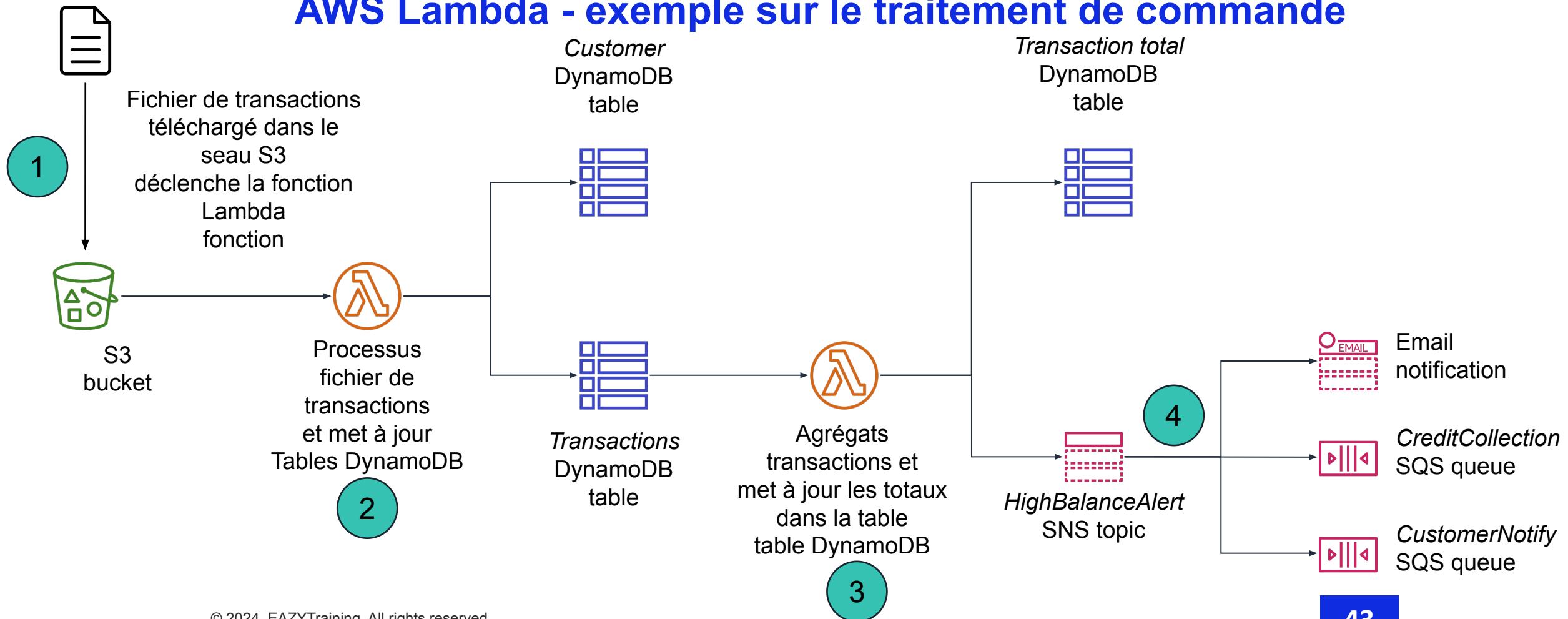
```
lambda.invoke(pullParams,  
function(error, data) {  
if (error) {  
prompt(error);  
} else {  
pullResults =  
JSON.parse(data.Payload);  
}  
});
```

AWS SDK for  
JavaScript

```
{  
iswinner: false,  
leftwheelimage: {s:  
'cherry.png'},  
midwheelimage: {s:  
'puppy.png'},  
rightwheelimage: {s:  
'robot.png'}
```

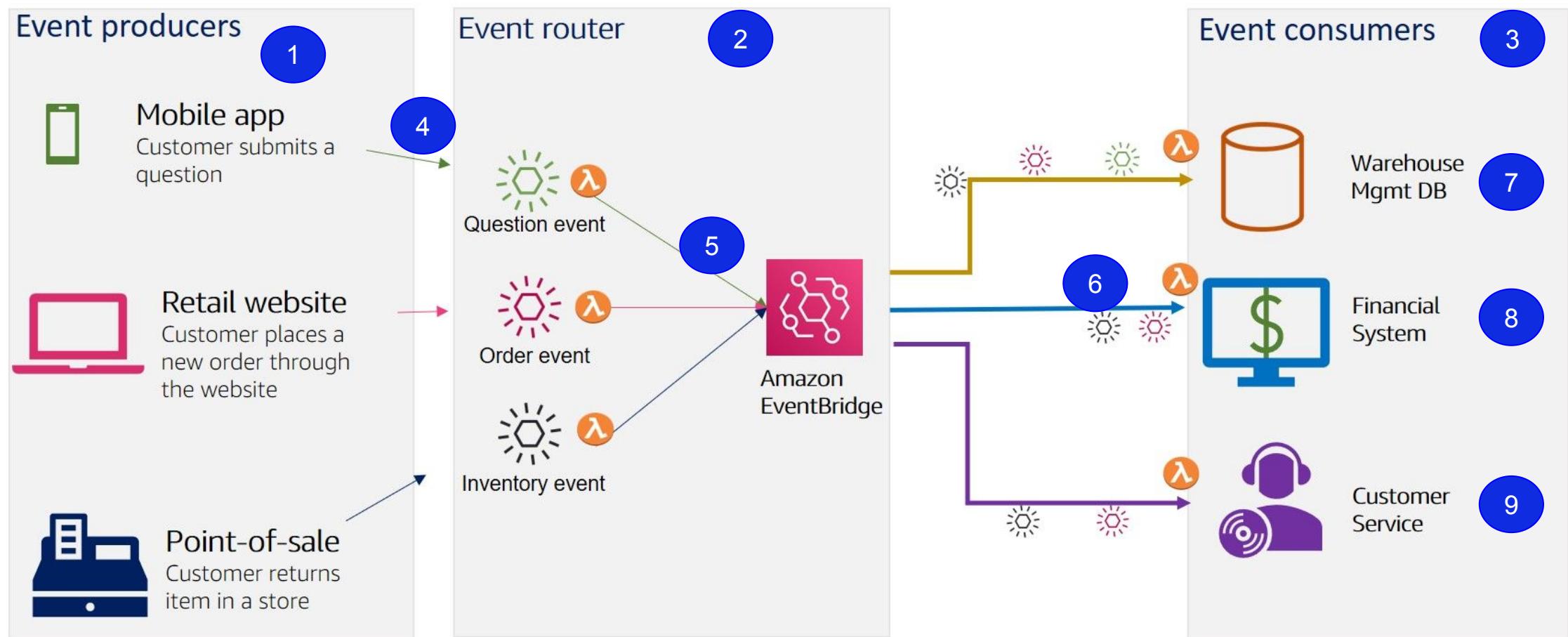


## AWS Lambda - exemple sur le traitement de commande





# AWS Lambda - Architectures événementielles





## Lambda Layers



- Permettre aux fonctions de partager facilement du code
  - Vous pouvez télécharger une couche une fois et la référencer dans n'importe quelle fonction.
- Promouvoir la séparation des responsabilités - Les développeurs peuvent itérer plus rapidement sur l'écriture de la logique métier.
- Permet de réduire le nombre de paquets à déployer
- Limites
  - Jusqu'à cinq couches
  - 250 MO



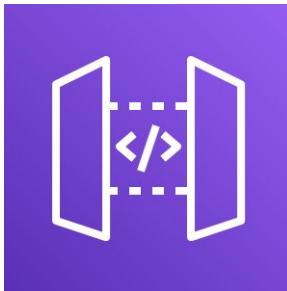
# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- **Amazon API Gateway**
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





## Amazon API Gateway

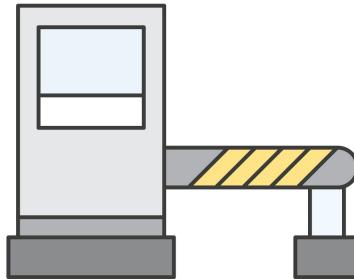


Amazon  
API  
Gateway

- Vous permet de créer, de publier, de maintenir, de surveiller et de sécuriser des API qui servent de points d'entrée vers des ressources dorsales pour vos applications
- Gère jusqu'à des centaines de milliers d'appels d'API simultanés.
- Peut gérer des charges de travail qui s'exécutent sur -
  - Amazon EC2
  - Lambda
  - Toute application web
  - Applications de communication en temps réel
- Peut héberger et utiliser plusieurs versions et étapes de vos API.



## Amazon API Gateway -sécurité



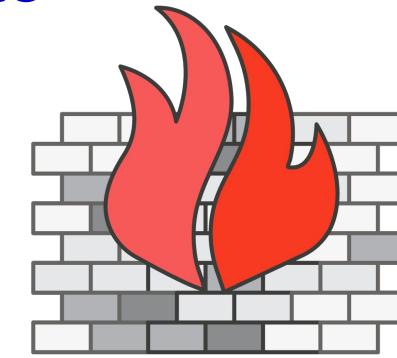
Exige une  
autorisation



Applique les  
politiques basées  
sur les  
ressources



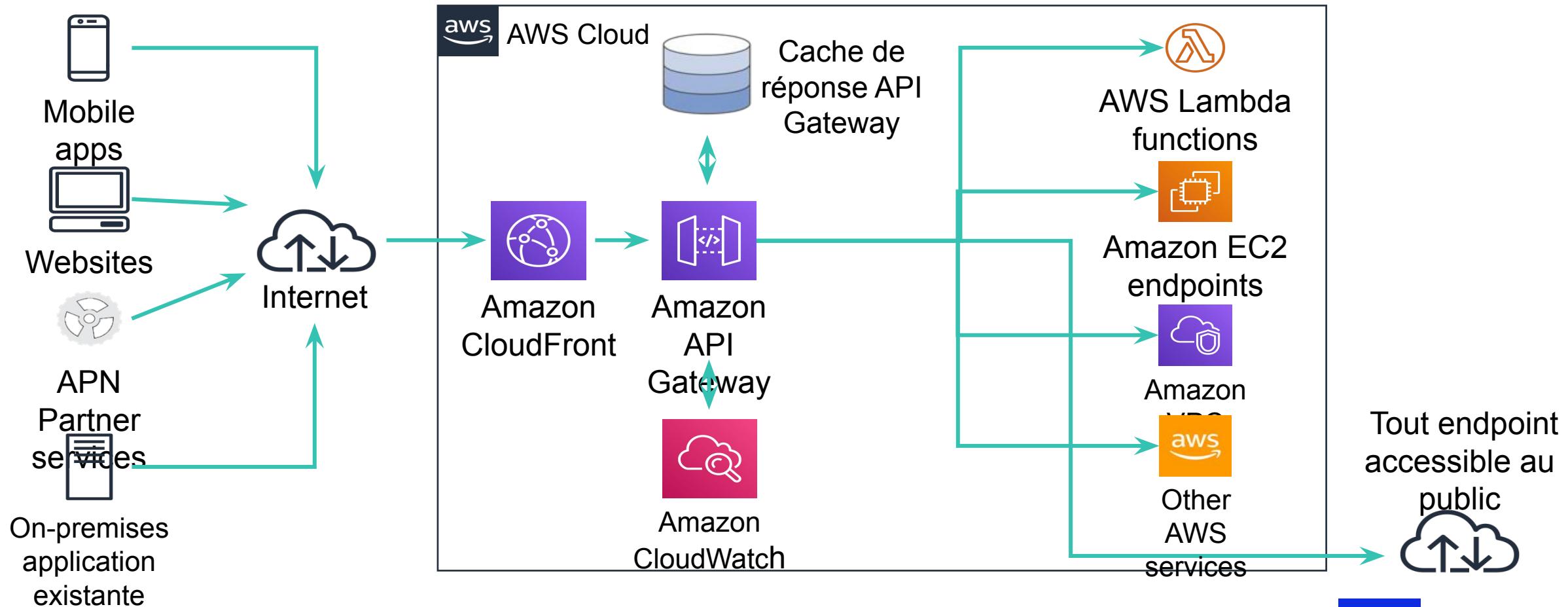
Restriction  
des  
paramètres



Protection des  
attaques de DDoS

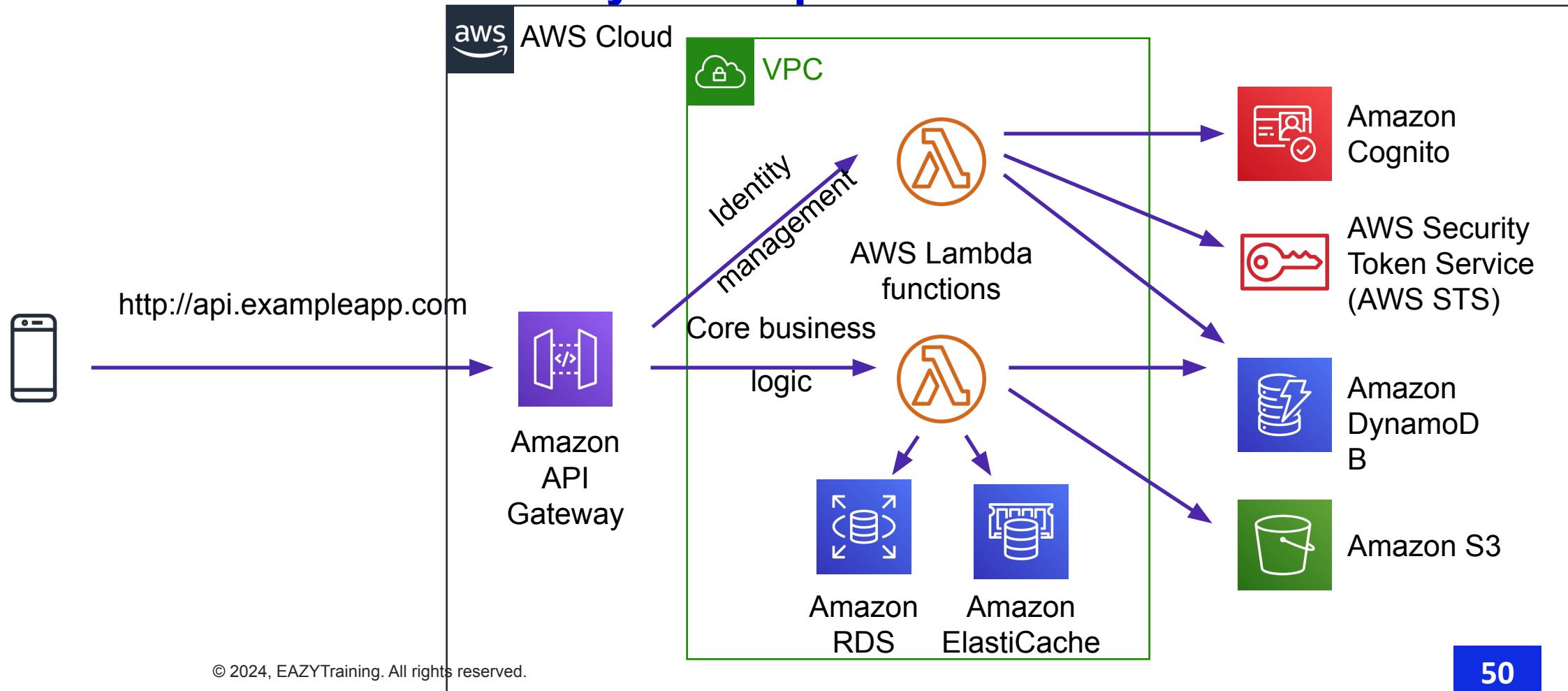


## Amazon API Gateway -exemple d'architecture





## Amazon API Gateway -exemple d'un backend mobile serverless





# Plan

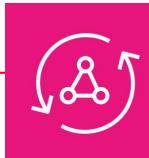
- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- **AWS AppSync**
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





# AWS AppSync

## Définition



AWS AppSync est un service sans serveur utilisé pour construire des API GraphQL avec des données en temps réel données en temps réel et des fonctionnalités caractéristiques.

Il remplace la fonctionnalité de Cognito Sync en fournissant une synchronisation des données hors ligne.

Il améliore les performances en fournissant des caches de données, des abonnements pour prendre en charge les mises à jour en temps réel des magasins de données côté client pour maintenir la synchronisation des clients hors ligne.

Il offre certains avantages par rapport à GraphQL, tels qu'un style de codage amélioré et une intégration transparente avec des outils modernes comme iOS et Android.



# AWS AppSync



L'interface AppSync offre une fonction d'API GraphQL en direct qui permet aux utilisateurs de tester et d'itérer sur les schémas GraphQL et les sources de données GraphQL.

En plus d'AppSync, AWS fournit un cadre Amplify qui permet de créer des applications mobiles et web à l'aide d'API GraphQL(interfaces de programmation).

**GraphQL** est un langage de données conçu pour permettre d'extraire des données des serveurs.



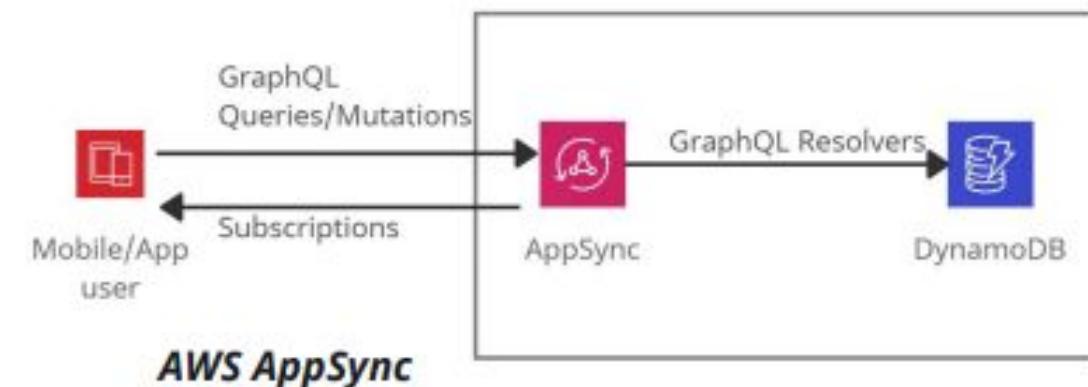
# AWS AppSync



**Requêtes** : Pour obtenir des données de l'API

**Mutations** : Pour modifier les données via l'API

**Abonnements** : Les connexions pour la diffusion en continu des données de l'API.



- Les différentes sources de données supporté par AppSync sont: Amazon DynamoDB, Amazon RDS, ElasticSearch, Lambda function, third party HTTP Endpoint



# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- **AWS SAM**
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**



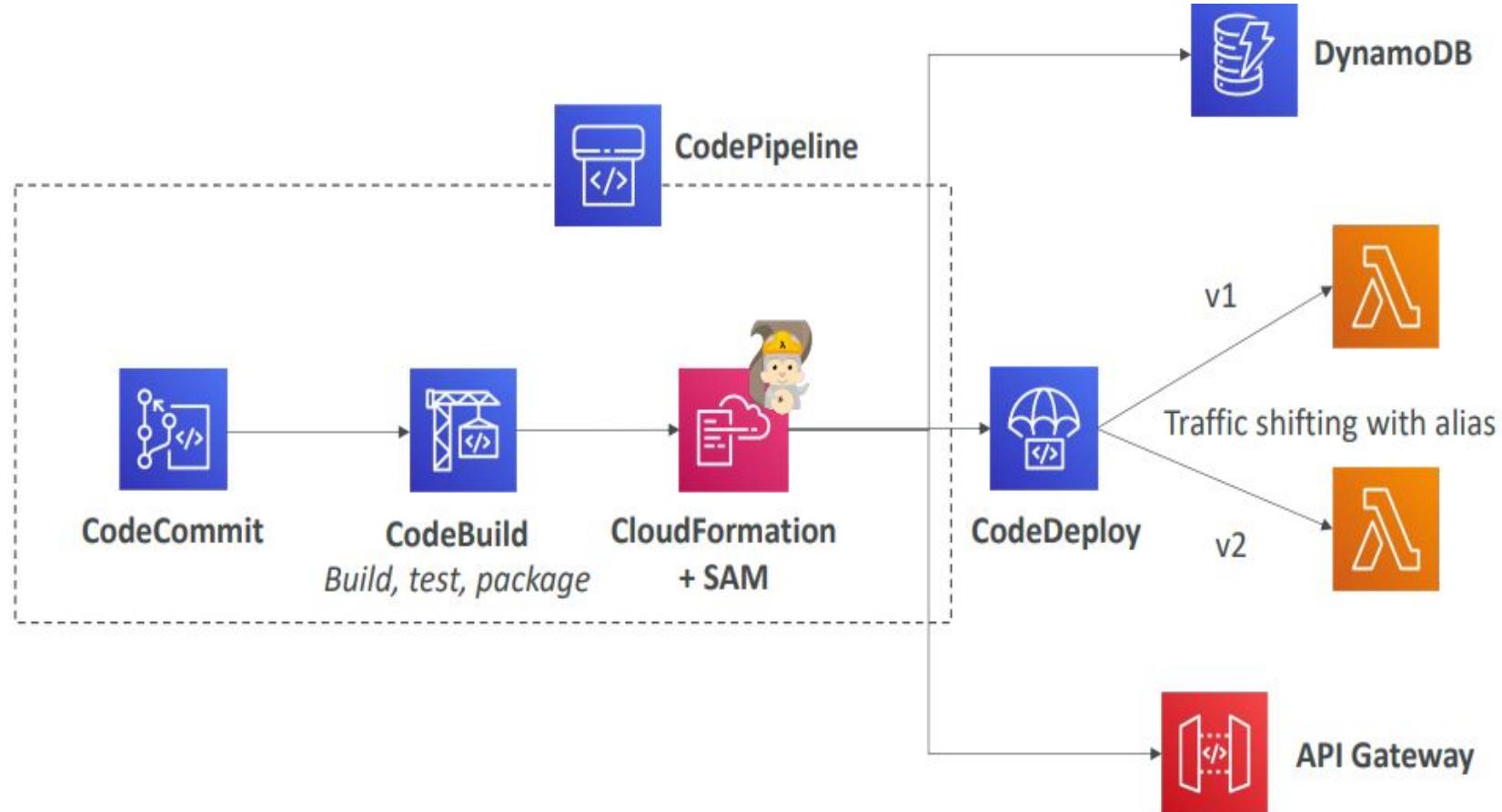


# AWS Serverless Application Model (SAM)

- **SAM = Modèle d'application sans serveur**
- Cadre pour le développement et le déploiement d'applications sans serveur.
- Toute la configuration est du code YAML. Exemples :
  - Fonctions Lambda (AWS::Serverless::Function)
  - Tables DynamoDB (AWS::Serverless::SimpleTable)
  - Passerelle API (AWS::Serverless::API)
  - StepFunction - Machine d'état (AWS::Serverless::StateMachine).
- SAM peut vous aider à exécuter Lambda, API Gateway, DynamoDB localement.
- **SAM peut utiliser CodeDeploy pour déployer les fonctions Lambda (transfert de trafic).**
- Tire parti de CloudFormation dans le backend



# AWS (SAM) -CI/CD architecture





# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- **Amazon Cloudfront**
- ElastiCache
- **Amazon Dax**





## Mise en cache des contenus avec Amazon CloudFront



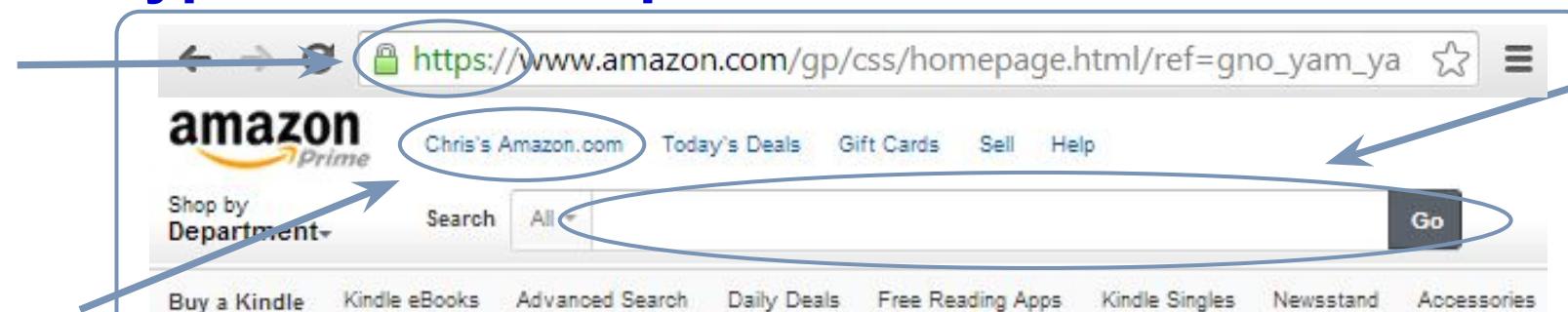
Amazon  
CloudFront

- Est le CDN mondial d'Amazon
- Est optimisé pour tous les cas d'utilisation de la diffusion, avec un cache à plusieurs niveaux par défaut et une grande flexibilité
- Fournit une couche supplémentaire de sécurité pour vos architectures
- Prend en charge les WebSockets et les méthodes HTTP ou HTTPS.



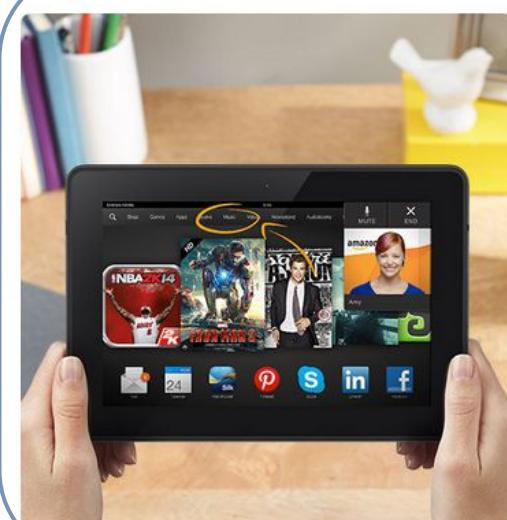
## Quel type de contenu pouvez-vous mettre en cache dans un

Sécurisé



User input

Dynamique



Image

Peuvent être mis en cache

Revolutionary on-device tech support

Exclusively on Kindle Fire HDX tablets—live on-device tech support from an Amazon expert is just a tap away with the new "Mayday" button

### Live Support with Mayday

**NEW**—Simply tap the "Mayday" button to be connected for free to an Amazon expert who can co-pilot you through any feature by drawing on your screen, walking you through how to do something yourself, or doing it for you—whatever works best. Mayday is available 24x7, 365 days a year, and it's free. Throughout the process, you will be able to see your Amazon Tech advisor live on your screen, but they won't see you. 15 seconds or less is the Mayday response time goal.

### Watch it in Action

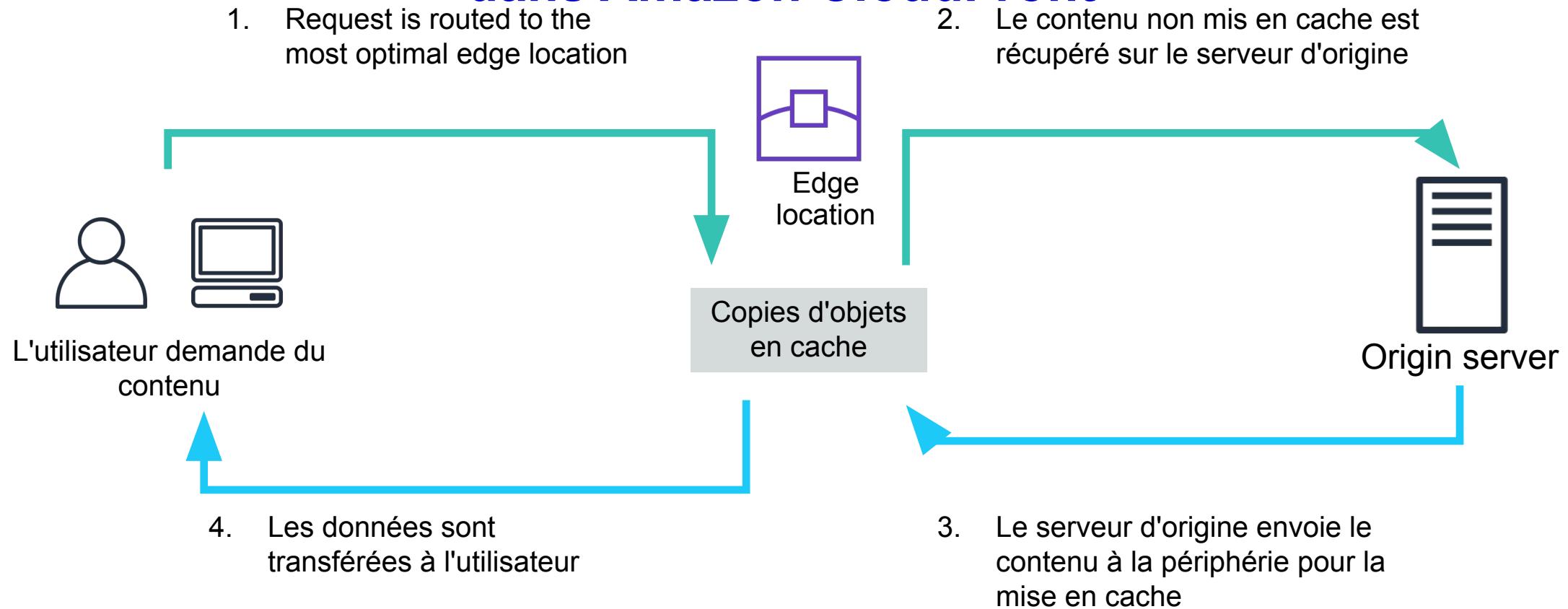
Kindle Fire HDX is easy to use right out of the box. But when you need extra assistance, the "Mayday" button is there. See how it works in these short commercials.

Video

Peuvent être mis en cache



## Amazon CloudFront -Comment fonctionne la mise en cache dans Amazon CloudFront





# Amazon CloudFront -Comment configuration d'une distribution CloudFront

1. Vous spécifiez le serveur d'origine



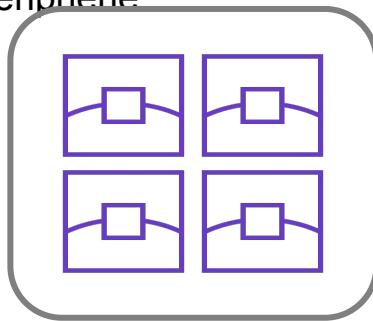
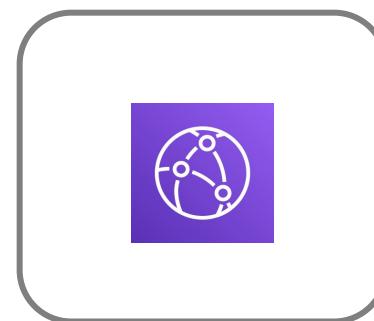
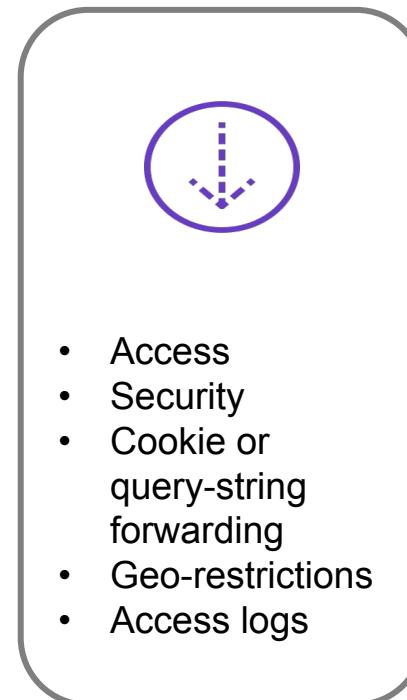
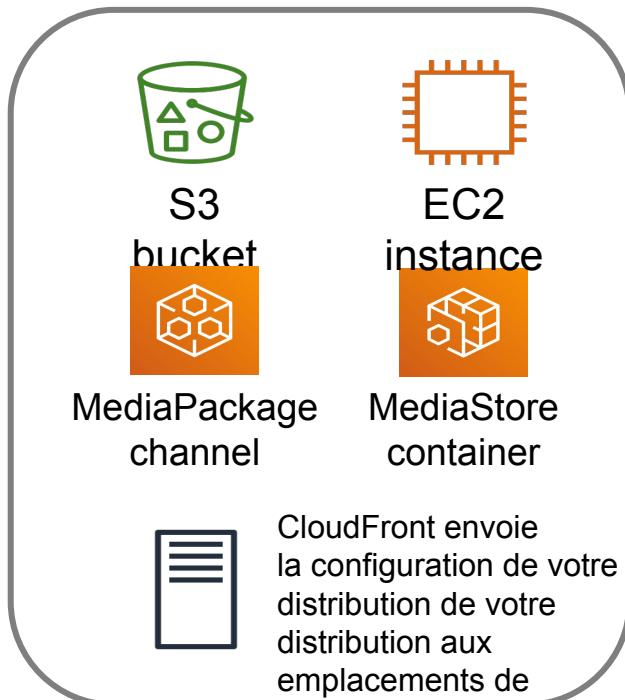
2. Configurez la distribution



3. Assignez un DNS à votre distribution



4. CloudFront envoie la configuration de votre distribution aux emplacements de périphérie



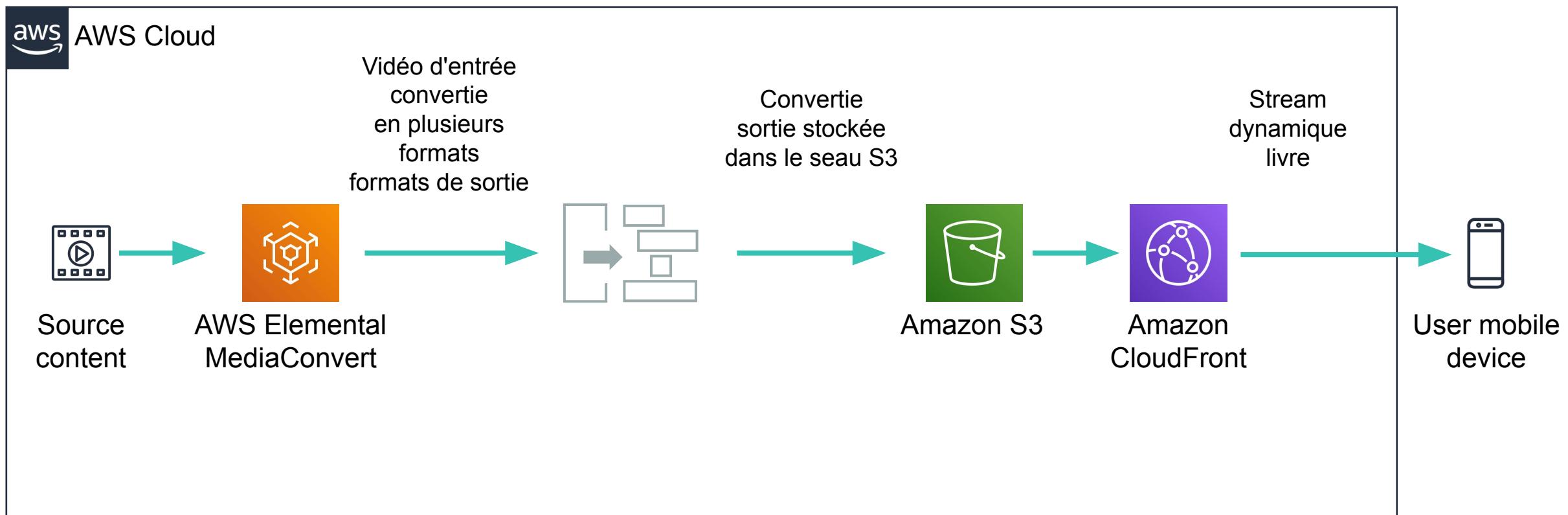


## Amazon CloudFront -Comment contenus expires

- Temps de vie (TTL) -
  - Période de temps fixe (période d'expiration)
  - Défini par vous
  - La requête GET vers l'origine depuis CloudFront utilise l'en-tête If-Modified-Since
- Changer le nom de l'objet -
  - Header-v1.jpg devient Header-v2.jpg
  - Le nouveau nom impose un rafraîchissement immédiat
- Invalider l'objet
  - Dernier recours : inefficace et coûteux



## Amazon CloudFront -exemple de streaming vidéo à la demande

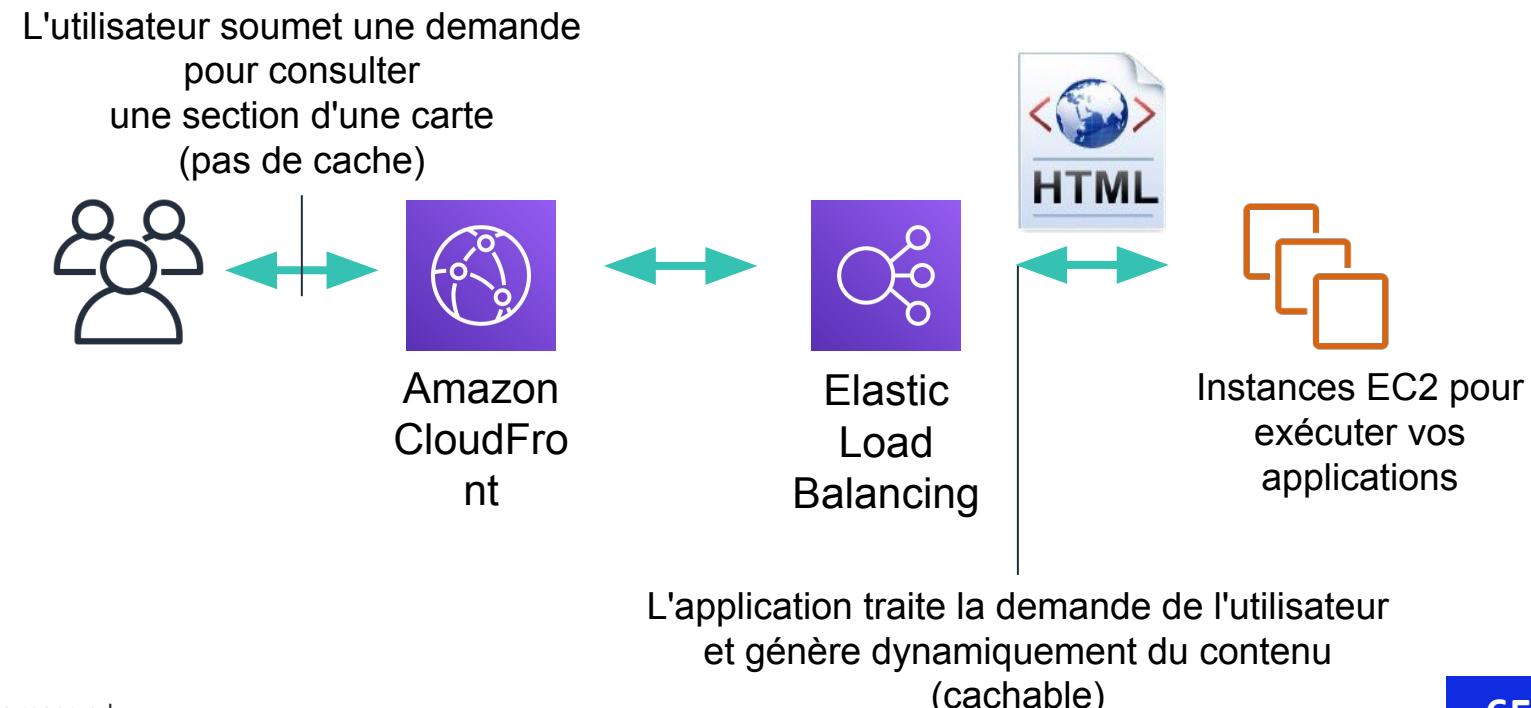




## Amazon CloudFront -exemple de contenus générés dynamiquement

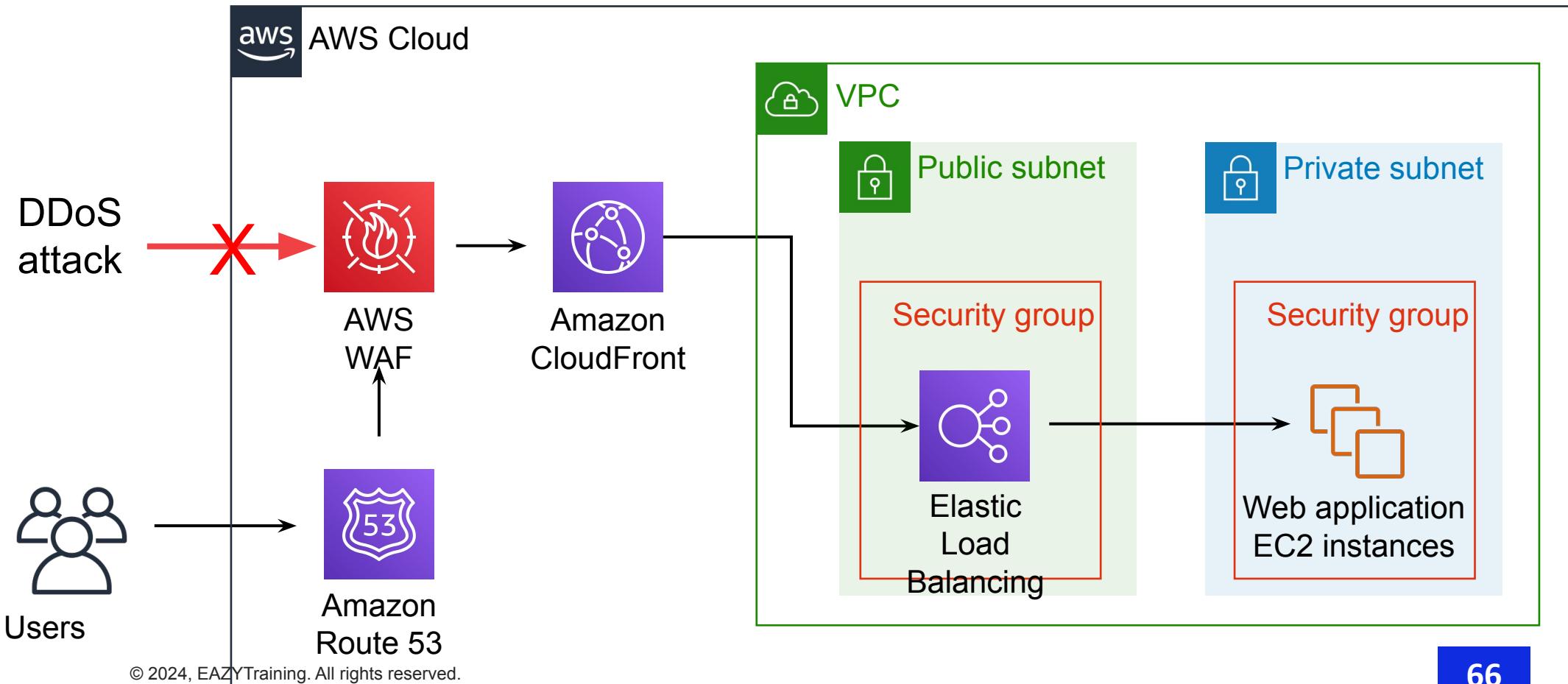
Cas d'utilisation : Tuiles de cartes

Problème : Besoin d'un temps de réponse plus rapide de la base de données





## Amazon CloudFront -exemple d'atténuation des attaques DDoS





# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- Amazon Dax





## Caching des bases de données: Amazon ElastiCache



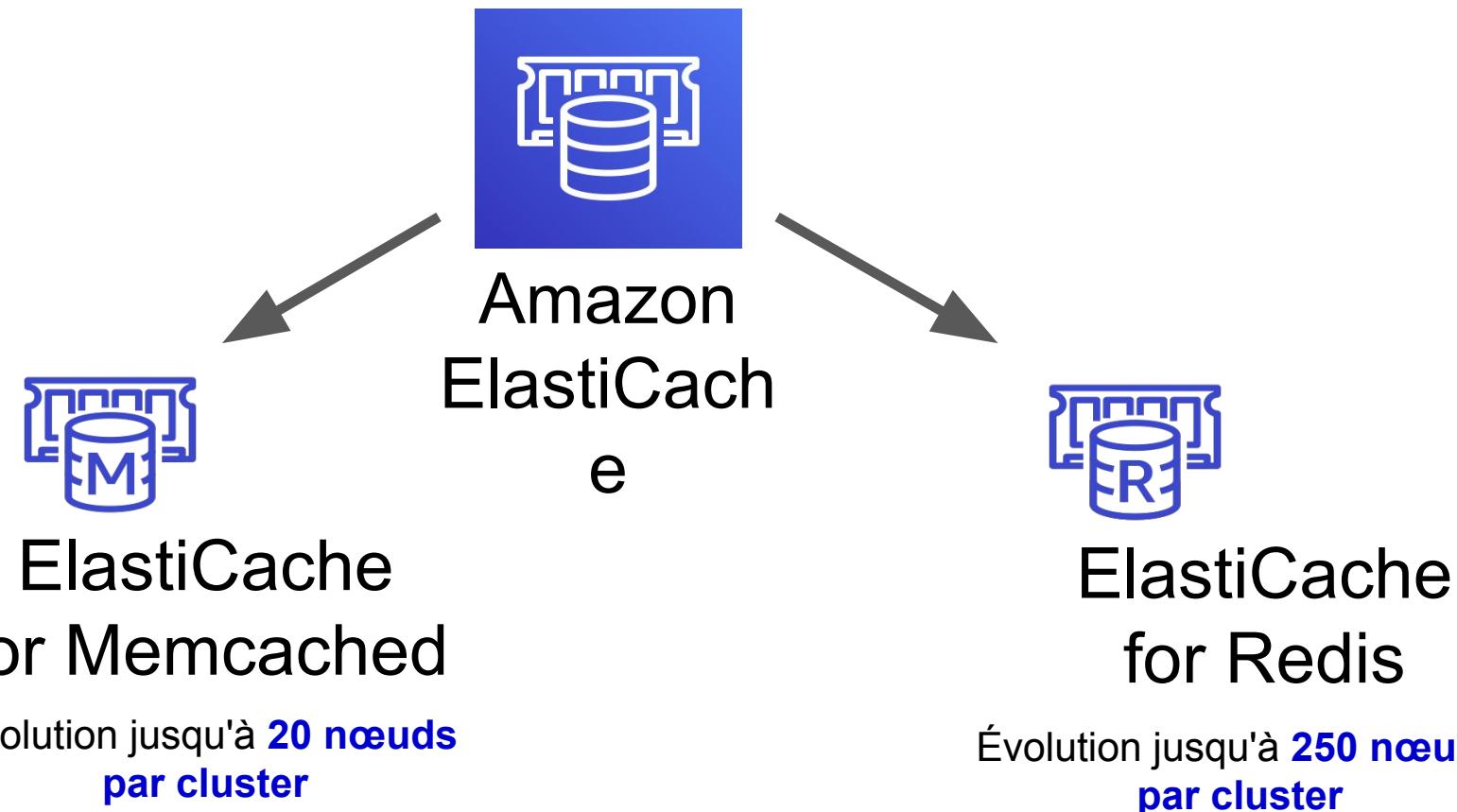
Amazon  
ElastiCache

ElastiCache fournit aux applications web un magasin de données en mémoire dans le cloud.

- stockage de données en mémoire dans le nuage.
- Fonctionne comme un magasin de données et un cache en mémoire
- Offre de hautes performances
- Est entièrement géré
- Est évolutif
- Prend en charge Redis et Memcached



## Amazon ElastiCache - Redis et Memcached



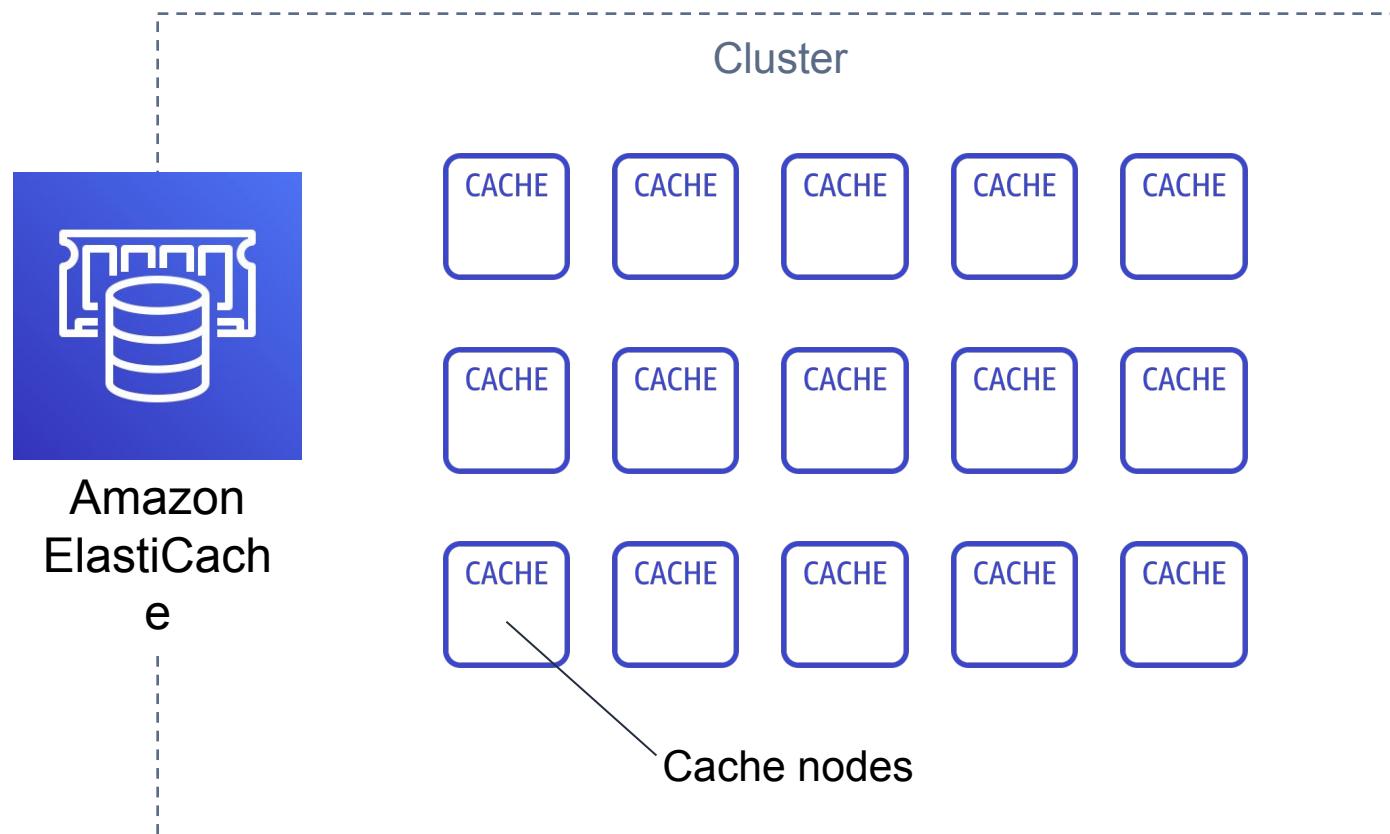


## Amazon ElastiCache - Redis versus Memcached

Feature	Memcached	Redis
Sub-millisecond latency	Yes	Yes
Ability to scale horizontally for writes and storage	Yes	No
Multi-threaded performance	Yes	No
Advanced data structures	No	Yes
Sorting and ranking datasets	No	Yes
Publish/subscribe messaging	No	Yes
Multi-AZ deployments with automatic failover	No	Yes
Persistence	No	Yes



## Amazon ElastiCache - composants

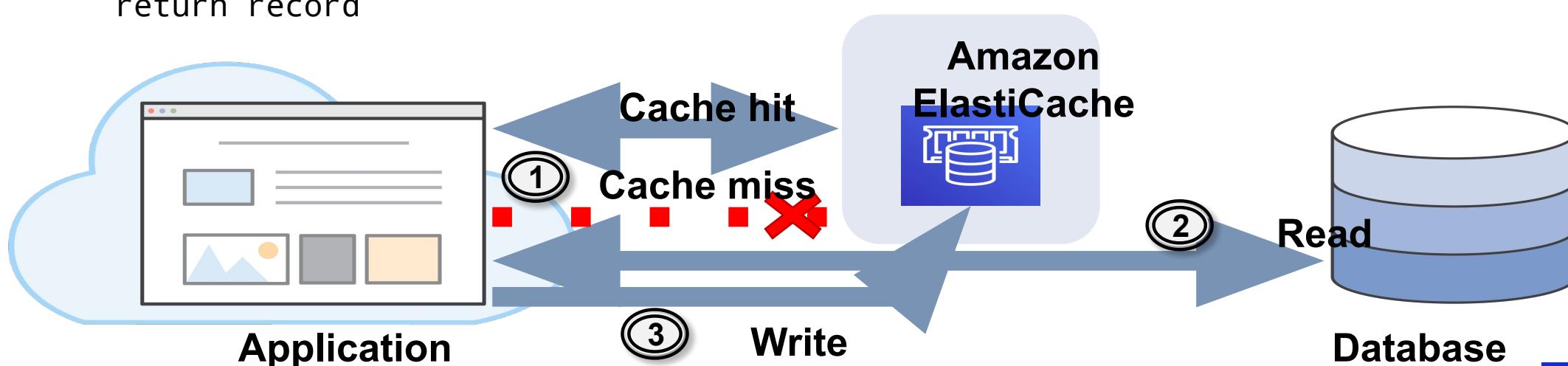


- Un nœud est le plus petit bloc d'un déploiement ElastiCache
- Chaque nœud a son propre nom DNS
- et son propre port
- Un cluster est un regroupement logique de
- un ou plusieurs nœuds



## Amazon ElastiCache - stratégies de caching: lazy loading

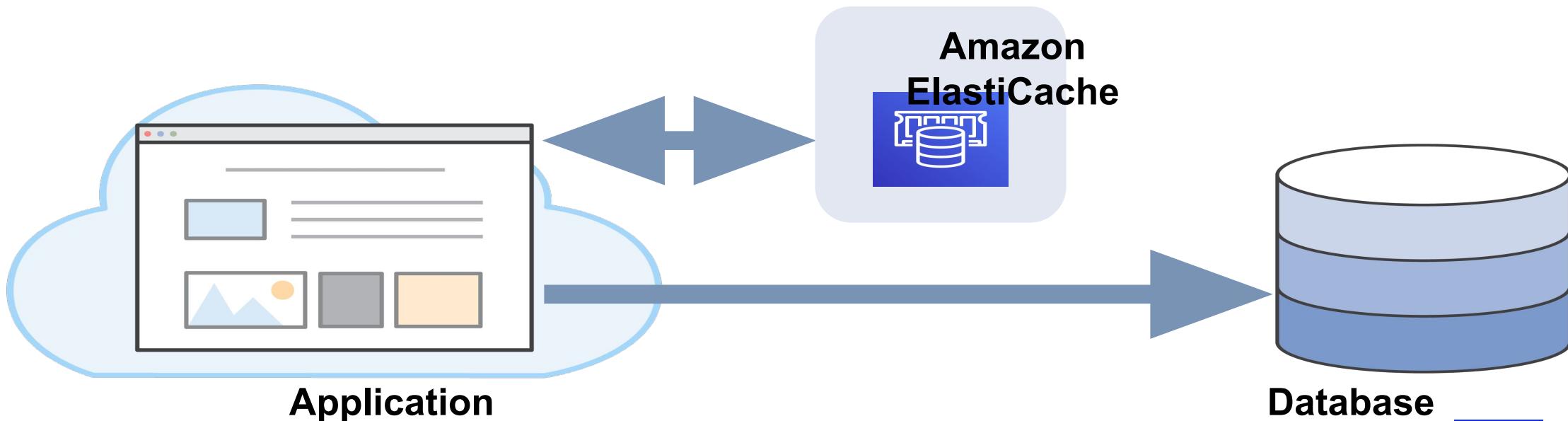
```
def get_user(user_id):
    # Check the cache
    record = cache.get(user_id)
    if record is None:
        # Run a DB query
        record = db.query("select * from users where id = ?", user_id)
        # Populate the cache
        cache.set(user_id, record)
    return record
```





## Amazon ElastiCache - stratégies de caching: Write-through

```
def save_user(user_id, values):
    # Save to DB
    record = db.query("update users...where id = ?", user_id, values)
    # Push into cache
    cache.set(user_id, record, 300) # TTL
    return record
```



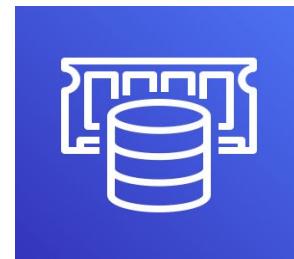


## Amazon ElastiCache - ajouter TTL

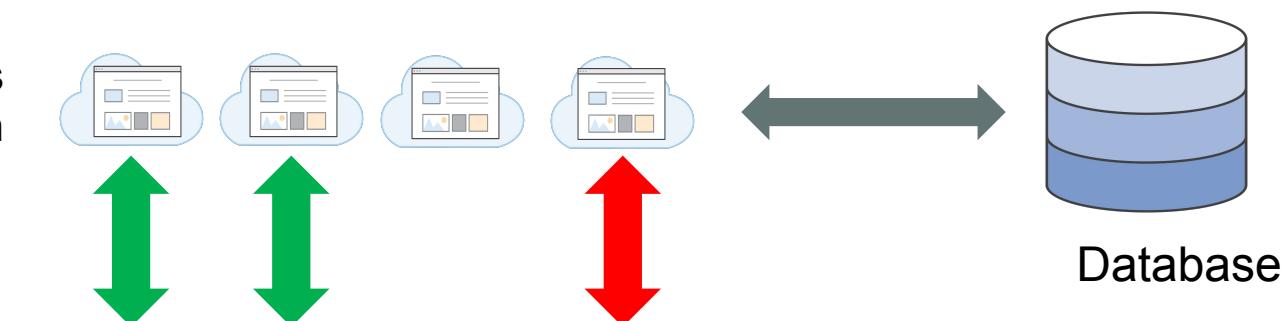
Serveurs d'application

Le TTL valeur ajoutée à chaque écriture de l'application.

1



Amazon  
ElastiCache



2

Après l'expiration du TTL, l'application interroge la base de données pour obtenir des données.



# Plan

- Vue globale de AWS Serverless
- Amazon EKS
- Amazon ECS
- Amazon ECR
- AWS Fargate
- Amazon X-Ray
- AWS Lambda
- Amazon API Gateway
- AWS AppSync
- AWS SAM
- Amazon Cloudfront
- ElastiCache
- **Amazon Dax**





# Utilisation de DynamoDB pour des informations sur l'état

Cas d'utilisation : application de jeu en ligne

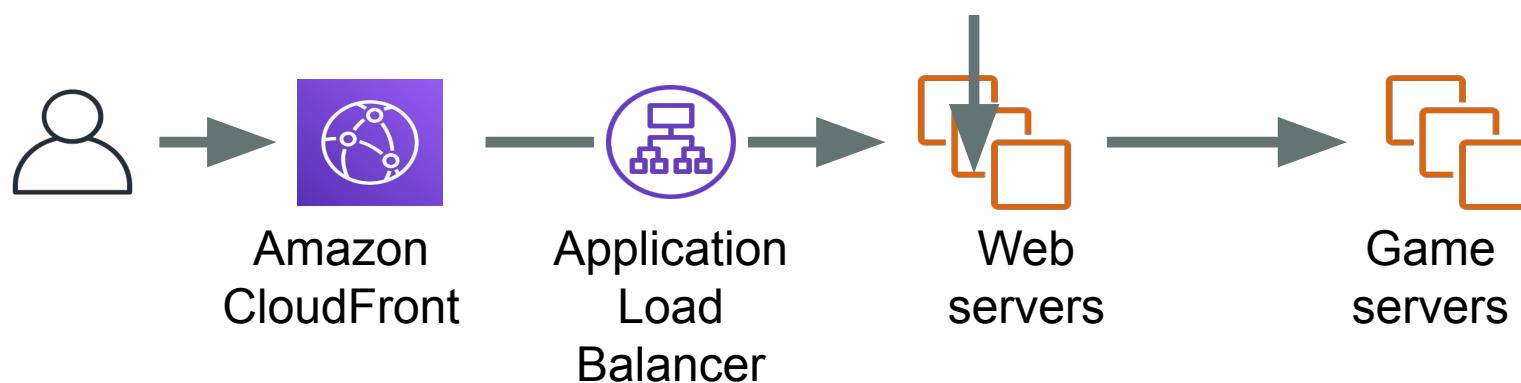
Problème : besoin d'un temps de réponse de base de données plus rapide.



Informations sur l'état de la session stockées dans Amazon DynamoDB



Temps de réponse à l'échelle de la milliseconde





## Amazon DynamoDB Accelerator (DAX)



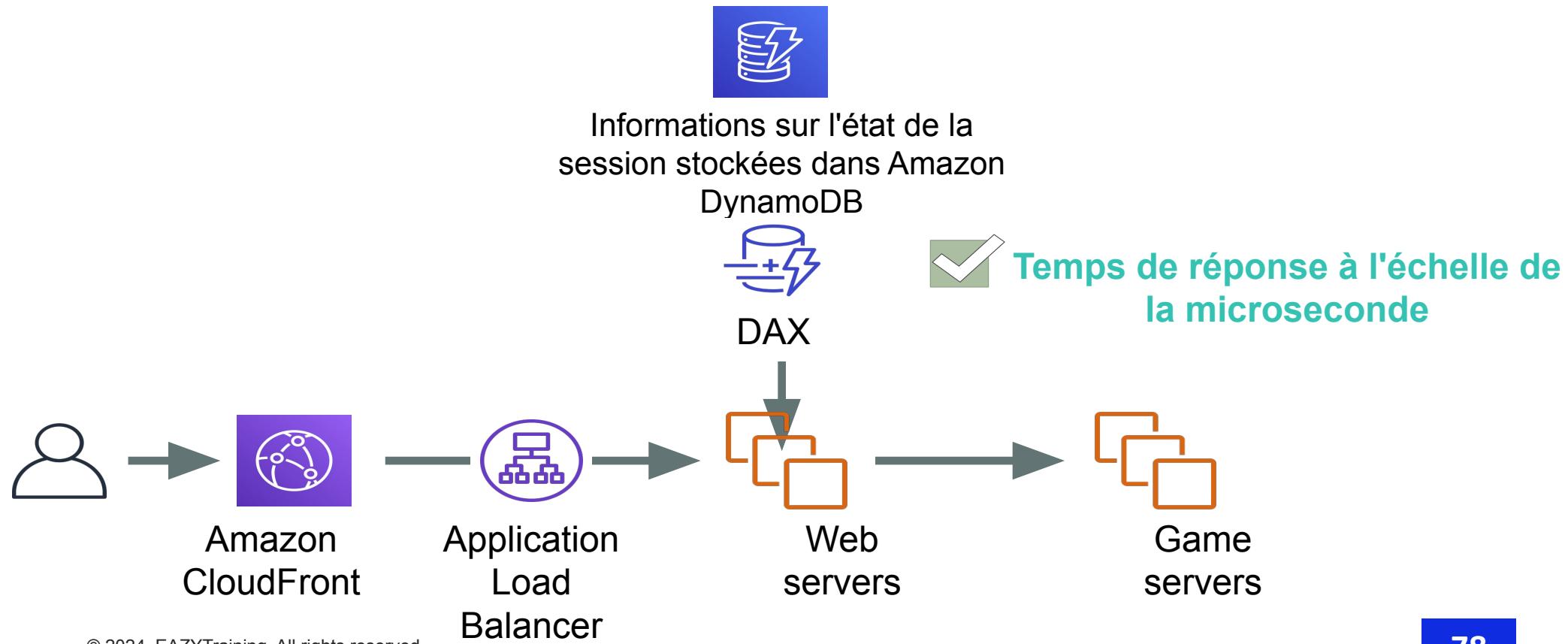
Amazon  
DynamoDB  
Accelerator

Cache en mémoire entièrement géré et hautement disponible pour  
DynamoDB

- Performances extrêmes (temps de réponse à l'échelle de la microseconde)
- Hautement évolutif
- Entièrement géré
- Intégré à DynamoDB
- Flexible
- Sécurisé



# Amazon DynamoDB DAX pour l'accélération de temps de réponse

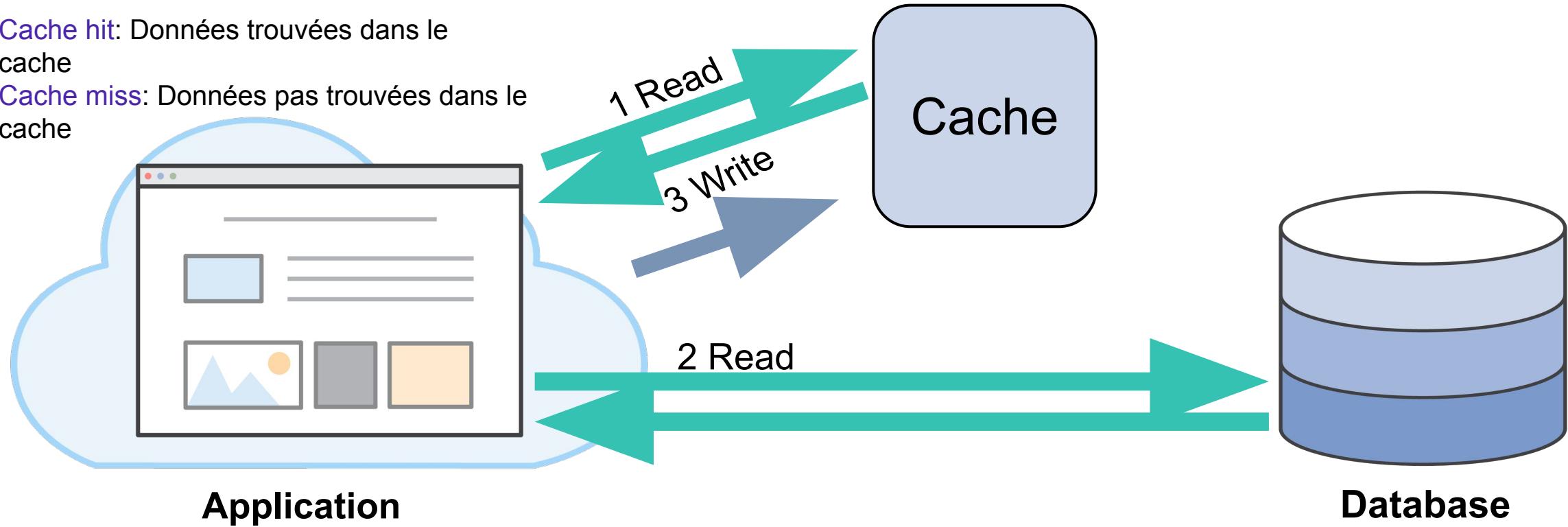




## Amazon DynamoDB DAX -cache à distance ou de côtés

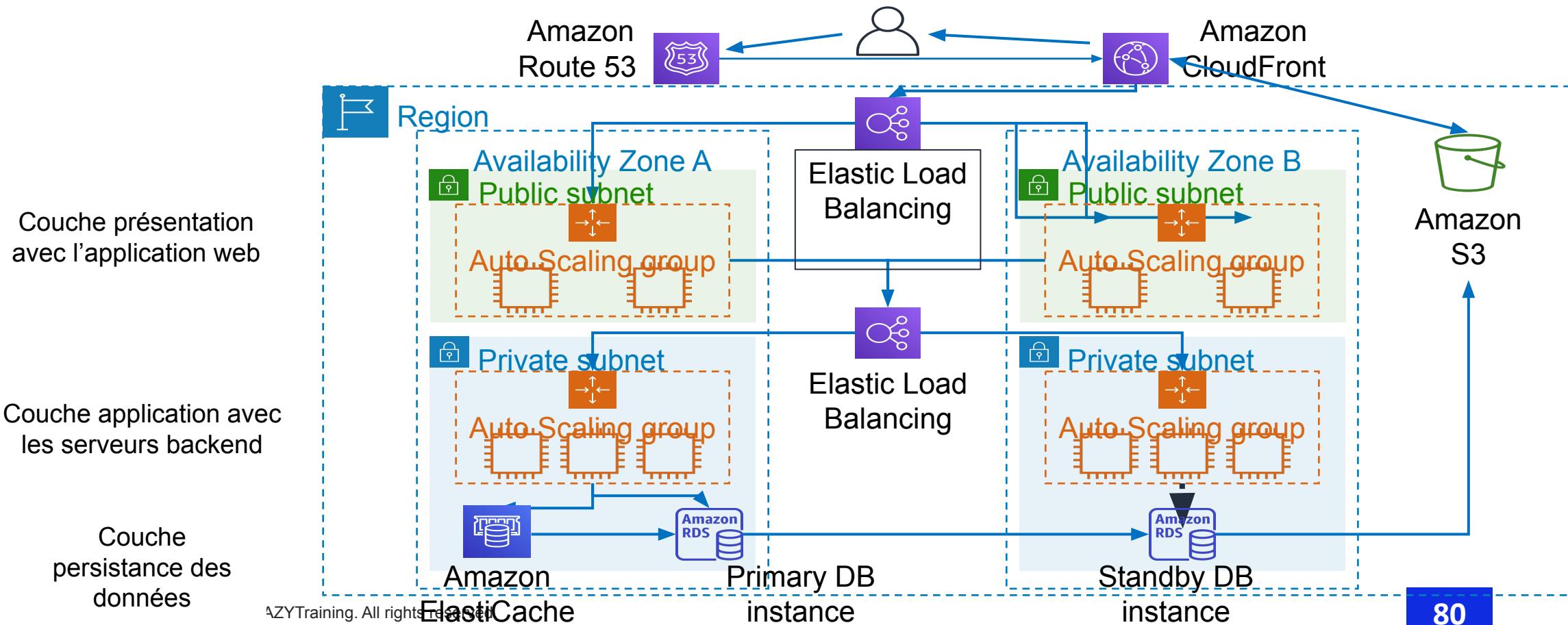
Cache hit: Données trouvées dans le cache

Cache miss: Données pas trouvées dans le cache





# Architecture 3-tiers avec application web





MERCI POUR VOTRE AIMABLE  
ATTENTION!



**Lahda Biassou Alphonsine**  
Ingénieure cloud et Formatrice