



# OC Pizza solution technique



# OC Pizza

## Choix de l'architecture.

Deux possibilités techniques peuvent être mise en œuvre pour l'application une version monolithique ou une version à base de microservices.

### Architecture Monolithique.

Toute l'application est implémentée dans un seul programme et fonctionne sur une seule machine. On a une grosse application complexe qui demande beaucoup de ressource pour démarrer et fonctionner. En outre l'application n'est pas flexible et ne peut pas répondre à un accroissement de la demande des utilisateurs. Elle n'est pas scalable. Si on a besoin de répondre à plus de demande on va devoir dupliquer l'application mais cela pose un autre problème pour la gestion des données.

On doit donc avoir au minimum un serveur sur lequel fonctionne l'applicatif et un autre pour gérer la base de données. On pourra ainsi dupliquer l'applicatif sur un autre serveur qui communiquera avec la même base de données. On accroît ainsi le problème d'utilisation de ressources car on duplique même des services que l'on n'a pas besoin comme la consultation de la liste des clients ou le lancement de la préparation d'une commande.

Il est aussi difficile de demander à son fournisseur de service internet d'allouer un nouveau serveur pour dupliquer notre application dans un temps record.

Le premier inconvénient de la configuration monolithique est l'utilisation massive de ressources qui doivent être multipliées pour répondre à la demande sous réserve de la passation d'un nouveau contrat avec le fournisseur.

Toute application doit être maintenue avec des mise à jour et c'est difficile de savoir où se trouve le problème dans une application volumineuse et complexe. On doit en outre l'arrêter et la redémarrer complètement ce qui n'est pas seulement une perte de temps mais aussi de profit.

La réflexion sur une application monolithique a soulevé de gros désavantages et montré que nous devons avoir au moins deux serveurs :

- Un pour la base de données.
- Un pour le programme.

La seconde approche est le découpage de l'application en plusieurs parties : les microservices.

## Architecture microservices.

Dans cette architecture on découpe l'application en services indépendant les uns des autres. Chaque microservice est développé, testé et déployé séparément des autres. Ils échangent des données avec les autres à travers les différentes APIs qu'ils exposent. Ils suivent les règles suivantes :

- Ils implémentent une seule fonctionnalité.
- Ils peuvent être écrit avec des langages différents.
- Ils sont faiblement couplés car séparé physiquement les uns des autres.
- Ils sont facilement testés, déployés et maintenus.

Les microservices sont petits comparé à une application monolithique et ils peuvent fonctionner sur un même serveur ou des serveurs très proches. L'accroissement du temps de réponse dû à leur séparation est négligeable par rapport à une architecture monolithique. En outre comme ils sont petits, ils répondent plus vite.

### *Scalabilité.*

Pour répondre à la demande d'un site internet il est plus facile de dupliquer un microservice qu'une application monolithique. En plus les contrats cher l'hébergeur permet d'avoir des configurations flexibles. En outre on peut séparer les services qui demandent plus de mémoire et ceux qui demandent plus de puissance de calcul. On n'a pas besoin de refaire un contrat avec l'hébergeur pour faire passer de 2 à 5 microservices.

### *Maintenance.*

Par sa définition un microservice est un programme petit donc facile à maintenir et à faire évoluer. Il est plus facile aussi de rechercher et d'isoler une erreur dans 100 lignes de code que dans 10000 lignes. On peut aussi en cas de problème facilement arrêter et relancer un microservice sans bloquer les autres. On peut aussi porter un microservice dans un autre langage sans que cela ne perturbe les autres.

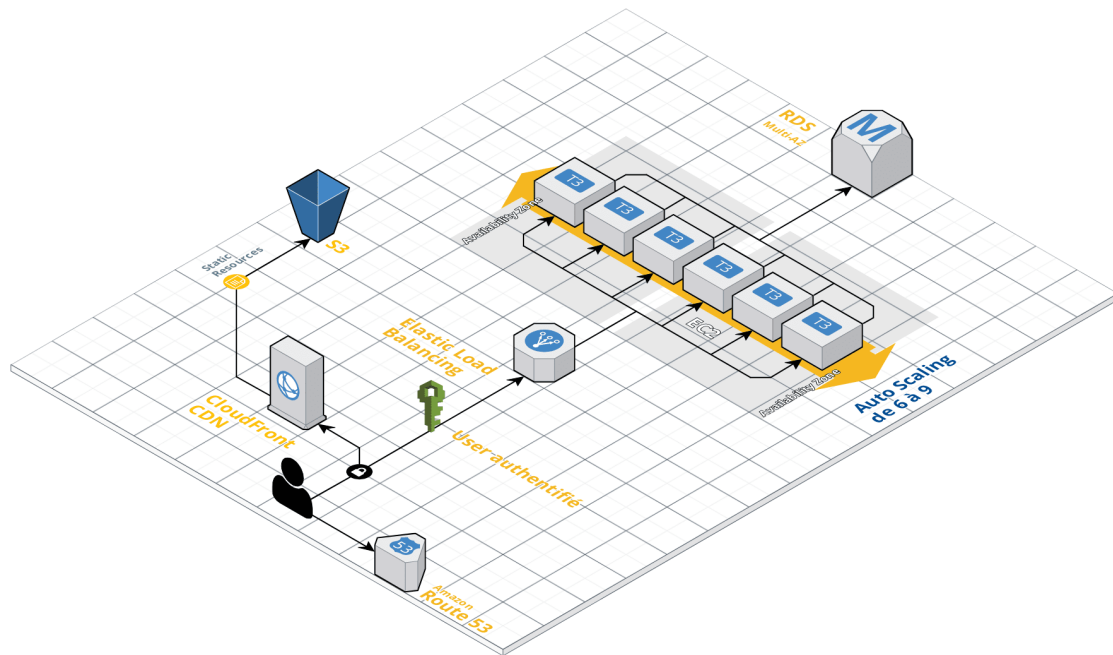
### *Encapsulation.*

On utilise des conteneurs comme Docker pour encapsuler les microservices avec les ressources dont ils ont besoin. Ils peuvent être écrit dans des langages différents et faire appel à des bibliothèques ou frameworks différents. En cas de changement de ressource d'un microservice, les autres ne seront pas impactés.

## Proxy.

Il est préférable d'utiliser un microservice proxy pour contrôler les requêtes qui sont envoyées à notre application et ainsi la prémunir des attaques malveillantes. Cela permet aussi de découpler le système de l'extérieur et permettre ensuite de le changer si nécessaire dans le futur sans perturber l'utilisateur.

## Représentation de l'architecture.



## Evaluation des coûts avec

### Configure Elastic Load Balancing [Info](#)

#### ☒ Équilibreur de charge d'application

L'équilibreur de charge d'application est le mieux adapté à l'équilibrage de charge du trafic HTTP et HTTPS et fournit un routage avancé des requêtes ciblé sur la fourniture d'architectures d'applications modernes, y compris les microservices et les conteneurs. Fonctionnant au niveau de la requête individuelle (couche 7), l'équilibreur de charge d'application achemine le trafic vers des cibles dans Amazon Virtual Private Cloud (Amazon VPC) en fonction du contenu de la requête.

#### ☐ Équilibreur de charge de réseau (NLB)

Network Load Balancer est mieux adapté pour l'équilibrage de charge du trafic TCP (Transmission Control Protocol), UDP (User Datagram Protocol) et TLS (Transport Layer Security) où des performances extrêmes sont requises. En fonctionnant au niveau de la connexion (couche 4), Network Load Balancer achemine le trafic vers les cibles au sein d'Amazon Virtual Private Cloud (Amazon VPC) et est capable de traiter des millions de requêtes par seconde tout en maintenant une latence très faible. Network Load Balancer est également optimisé pour gérer les pointes soudaines et les modèles de trafic changeants.

#### ☐ Équilibreur de charge classique (CLB)

Classic Load Balancer fournit un équilibrage de charge de base sur plusieurs instances Amazon EC2 et fonctionne à la fois au niveau de la demande et de connexion. Classic Load Balancer est destiné aux applications qui ont été construites dans le réseau EC2-Classique.

#### Région

EU (Paris)


#### Paramètres de service [Informations](#)

Nombre d'équilibreurs de charge d'application

1

► Show calculations


### Cloudcraft.

Budget for OC Pizza Architecture 

MONTHLY

EFFECTIVE RATE

EUR

 EXPORT

Compute **€72,74** / mo ▼

Storage **€0,22** / mo ▼

Networking **€28,10** / mo ▼

Database **€49,36** / mo ▼

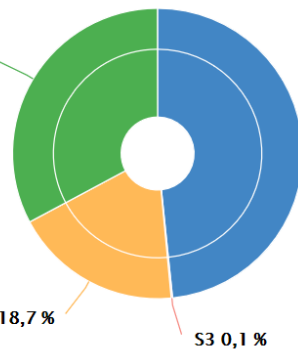
**Total €150,42** / mo

RDS 32,8 %

EC2 48,4 %

ELB 18,7 %

S3 0,1 %



S3	Region EU-WEST-3	Count 1							
Volume Type	Storage (GB)	PUT/COPY/POST / mo (K)	GET and other / mo (K)						
Standard	10	0	0						€0,22 / mo

RDS	Region EU-WEST-3	Count 1							
Role	Engine	Instance type	Size	Deployment option	Billing option				
Master	MySQL	T3 - Burst	small	Multi-AZ	On-Demand 100%				€49,36 / mo
Intel Skylake E5 2686 v5 (2.5 GHz)   2 vCPU   2 GiB memory   EBS Only   Low to Moderate networking performance									

ELB	Region EU-WEST-3	Count 1							
Type	LCUs								
Application	2								€28,10 / mo
50 New conn. / sec.   6000 Active conn. / min.   4.44 Mbps Bandwidth									

EC2	Region EU-WEST-3	Count 6							
Platform	Instance type	Size		Billing option					
Linux	T3 - Burst	small		Reserved 1Y/No Upfront/Std.					€65,08 / mo
Intel Skylake E5 2686 v5 (2.5 GHz)   2 vCPU   2 GiB memory   EBS only   Low to Moderate networking performance									

EC2	Region EU-WEST-3	Count 3							
Platform	Instance type	Size		Billing option					
Linux	T3 - Burst	small		Spot Market Rate, 16.666666666666664%					€7,66 / mo
Intel Skylake E5 2686 v5 (2.5 GHz)   2 vCPU   2 GiB memory   EBS only   Low to Moderate networking performance									

## Décomposition de l'application.

On scinde l'application en microservices qui sont peu couplés les uns aux autres. Suivant la montée en charge on pourra cibler les APIs à dupliquer pour pouvoir répondre à la demande. On peut faire un premier découpage en dédiant un microservice par package de la spécification fonctionnelle.

### Microservice commande.

C'est la partie utilisée pour la prise de commandes soit par le Client ou l'Employé. C'est lui aussi qui permet de faire évoluer le statut au cours de la vie d'une commande suivant les intervenants. C'est celui qui sera dupliqué pour répondre à la demande croissante des clients.

### Microservice stock.

C'est par ce microservice que toutes les modifications et consultations de stock passent. Il modifie la base de données du stock lors de la validation d'une commande où la réception d'une commande fournisseur. C'est lui aussi qui permet de savoir si la quantité d'un produit est suffisante pour être vendu.

### Microservice client.

C'est par lui que transitent toutes créations, modifications, consultations et suppressions de client. Il renvoie les données de livraison du client lors de la création d'une commande. Il permet d'accéder à des informations sur les anciennes commandes des clients à but statistique ou marketing.

### Microservice gestion.

Il regroupe tous les indicateurs et les informations sur les ventes, les coûts de revients, les profits, les pertes pour faire des statistiques. Il permet aussi d'effectuer de la gestion administrative du magasin avec la production de documents pro-formats.

### Microservice authentification.

C'est le service indispensable pour gérer les accès au système pour les Clients et les Employés d'OC Pizza.

## Description de l'architecture

Le choix du prestataire s'est orienté vers Amazon qui offre de nombreux services et outils. Il permet en plus de travailler par zone géographique dans le but de développer progressivement l'activité au niveau mondial.

Le site **aws.amazon.com** nous permet de calculer les coûts de déploiement sur le cloud d'Amazon suivant les options choisies.

### *Base de données.*

Le service **Amazon RDS (Relational Database Service)** permet de configurer les besoins en base de données en utilisant les plus grands **SGBD-R** mais aussi des services de migration et de réplication. Le service **RDS** permet d'accompagner la croissance de notre base de données et la faire évoluer facilement.

La gestion du service se fait soit par la console **RDS Management Console**, par la ligne de commandes ou par des appels **API**. On spécifie les détails de notre instance **MySQL** sur **AWS**.

- On prend l'option **Multi-AZ** pour avoir une réplication synchrone de la base dans une autre zone par sécurité. Cela augmente la latence des transactions mais les données sont protégées. On pourra toujours ajouter par la suite des réplications en lecture de la base pour accroître la capacité de réponse de la base de données.
- Une instance de type **db.t3.small** suffit au départ pour notre base de données. On pourra par la suite augmenter la taille si nécessaire.
- On prend donc une instance qui est en fait constituée d'une base de données principale et une secondaire de secours.



**Spécifications de l'instance MySQL** Informations

☐ **Standard (Mono-AZ)**  
Déploiement mono-AZ, Amazon RDS provisionne une base de données dans une zone de disponibilité.

☒ **Multi-AZ**  
Déploiement multi-AZ, Amazon RDS provisionne et gère un réplica de secours synchrone dans une autre zone de disponibilité.

Type d'instance

Q db.t3.small X

**db.t3.small**

Coût horaire à la demande	Tarif horaire d'une instance réservée (1 an, aucun coût initial)	vCPU
0.076 USD	0.0535 USD	2
Mémoire 2 GiB		

Quantité

1

- On réserve l'instance pour une année pour réduire les coûts et permettre de réajuster au bout d'un an de production le choix d'hébergement de la base de données.
- On pourra faire une réservation sur 3 ans quand la configuration sera stable et en période de non expansion du nombre de magasin.
- Pour ne pas accroître les coûts de départ, on coche l'option "Aucun frais initiaux".

**Stratégie de tarification** Informations

Modèle de tarification

- ☒ Instances réservées standard  
☐ Instances à la demande

Condition de réservation

- ☒ 1 an  
☐ 3 ans

Option de paiement

- ☒ Aucun frais initiaux  
☐ Frais initiaux partiels  
☐ Tous les frais initiaux

► Show calculations

- On utilise un stockage de type Solid State Disk qui sont plus performant que des disques mécaniques.
- Une capacité de 30Go est largement suffisante pour contenir la base pour plusieurs années.

### Stockage Informations

Saisissez la quantité de stockage désirée pour chaque instance.

Stockage pour chaque instance RDS

SSD à usage général (gp2)

Quantité de stockage

30

Go

► Show calculations

- Le coût total pour la base de données seule est de 47,03\$ soit 42,44€.

### Amazon RDS for MySQL estimate

Tarification du stockage (monthly)	7.98 USD
Coût mensuel des instances réservées MySQL Amazon RDS (monthly)	39.05 USD
<b>Total monthly cost:</b>	<b>47.03 USD</b>

### Application.

On a besoin d'un serveur pour faire fonctionner les microservices. On choisit un contrat flexible qui permet d'accroître le nombre d'instance de 2 à 5 pour répondre à la demande aux heures de pointes. On utilise le service **Amazon EC2** (Elastic Cloud) et un contrat sur une année pour permettre de le modifier si nécessaire après une année de fonctionnement.

- On utilise le système d'exploitation **Linux** pour exécuter les instances de nos microservices.
- On paramètre 6 instances par défaut qui fonctionnent 24/24 et 7/7.
  - 1 x API Authentification.
  - 1 x API Gestion infocentre.
  - 1 x API Stock.
  - 1 x API Clients

- 2 x API Commande qui est plus sollicité.
- On rajoute 3 instances pendant 4 heures pour les heures de fortes activités le soir pour répliquer les API Commande.

## Paramètre [Informations](#)


### Système d'exploitation


Choisissez le système d'exploitation avec lequel vous souhaitez exécuter les instances Amazon EC2.


Linux


## Workload [Informations](#)

Sélectionnez le graphique qui représente le mieux votre workload mensuel. Sélectionnez Custom (Personnalisé) pour définir les paramètres manuellement.

☐ Utilisation constante  


☒ Pic de trafic quotidien  


☐ Pic de trafic hebdomadaire  


☐ Pic de trafic mensuel  


▼ Daily spike pattern

Remove pattern

### Workload days

Select days for your workload pattern.

☒ Sunday ☒ Monday ☒ Tuesday ☒ Wednesday ☒ Thursday ☒ Friday ☒ Saturday

### Baseline

Enter the minimum number of instances that you need as a baseline for your workload.

6

### Peak

Enter the maximum number of instances that you need at the peak of your workload.

9

### Duration of peak (hours, minutes)

Enter the amount of hours and minutes your instances are running at peak.

4

0

- On utilise des instances **t3a.small** qui suffisent en capacité de mémoire et de cpu pour nos microservices.

**EC2 Instances (3)**  
Selected Instance: t3a.small

Q small X

Any vCPUs ▾ Any Memory (GiB) ▾ Any Network Performance ▾ Any storages ▾

☒ Show only current generation instances. < 1 > ⓘ

	Instance name ▲	Memory ▾	vCPUs ▾	Network Perf... ▾	Storage ▾	On-Demand... ▾	Reserved 1yr NoUpfront Ho... ▾
<input type="radio"/>	t2.small	2 GiB	1	Low to Moderate	EBS only	0.0264	0.0184
<input type="radio"/>	t3.small	2 GiB	2	Low to Moderate	EBS only	0.0236	0.0167
<input checked="" type="radio"/>	t3a.small	2 GiB	2	Low to Moderate	EBS only	0.0212	0.015

- On prend l'option de coût optimisé et une réservation de 1 an pour pouvoir modifier la configuration après une années d'exploitation.

**Stratégie de tarification** Informations

Modèle de tarification

☒ **Coût optimisé**  
Combinaison de la tarification Réserve et à la demande pour la meilleure valeur ajoutée et les meilleures performances.

☐ **Toutes réservées**  
Les instances réservées vous permettent de bénéficier d'une remise conséquente (jusqu'à 75 %) par rapport aux tarifs des instances à la demande.

☐ **À la demande**  
Payez pour la capacité de calcul par heure ou par seconde selon les instances que vous exécutez.

☐ **Spot**  
Les instances Spot Amazon EC2 vous permettent de demander une capacité de calcul Amazon EC2 non utilisée en bénéficiant d'une réduction allant jusqu'à 90 % par rapport aux tarifs à la demande.

**Reservation options (12)**

	Reservation term ▲	Convertible instances ▾	Upfront payment ▾	Upfront cost (USD) ▾	Monthly cost (USD) ▾
<input type="radio"/>	1 Year	Yes	None	0.00	83.51
<input type="radio"/>	1 Year	Yes	Partial	432.00	43.65
<input checked="" type="radio"/>	1 Year	No	None	0.00	73.44
<input type="radio"/>	1 Year	No	Partial	378.00	38.84

- 30Go pour chaque instance des **EC2** suffit largement pour contenir le système d'exploitation et un microservice.
- On utilise l'option **EBS** (Elastic Block Storage) pour pouvoir mutualiser le stockage des instances **EC2**.

**Amazon Elastic Block Storage (EBS)** [Informations](#)

Joindre des volumes de stockage de blocs persistants pour vos instances Amazon EC2

**Calculating EBS snapshots**[Learn more](#) on how EBS snapshot prices are calculated.**Stockage pour chaque instance EC2**

Choisissez le système d'exploitation avec lequel vous souhaitez exécuter les instances Amazon EC2.

SSD à usage général (gp2)

Quantité de stockage

30

Go

Fréquence d'instantané

2 fois par jour

Quantité modifiée par instantané

3

Go

[► Show calculations](#)

- Le coût total pour les microservices est de 137,31\$ soit 123,92€.

**Amazon EC2 estimate**

Tarification Amazon Elastic Block Storage (EBS) (monthly)	63.87 USD
Coûts des instances à la demande Amazon EC2 (monthly)	7.74 USD
Coût des instances réservées Amazon EC2 (monthly)	65.70 USD
<b>Total monthly cost:</b>	<b>137.31 USD</b>
<b>Total upfront cost:</b>	<b>0.00 USD</b>

## Load balancer.

On utilise **Amazon ELB** (Elastic Load Balancer) pour répartir la charge sur les différentes instances des microservice, notamment entre les API Commande. Son accès est sécurisé par le service **Amazon IAM** (Identity and Access Management) qui est compris dans le package.

## Configure Elastic Load Balancing [Info](#)

☒ **Équilibreur de charge d'application**  
L'équilibreur de charge d'application est le mieux adapté à l'équilibrage de charge du trafic HTTP et HTTPS et fournit un routage avancé des requêtes ciblé sur la fourniture d'architectures d'applications modernes, y compris les microservices et les conteneurs. Fonctionnant au niveau de la requête individuelle (couche 7), l'équilibreur de charge d'application achemine le trafic vers des cibles dans Amazon Virtual Private Cloud (Amazon VPC) en fonction du contenu de la requête.

☐ **Équilibreur de charge de réseau (NLB)**  
Network Load Balancer est mieux adapté pour l'équilibrage de charge du trafic TCP (Transmission Control Protocol), UDP (User Datagram Protocol) et TLS (Transport Layer Security) où des performances extrêmes sont requises. En fonctionnant au niveau de la connexion (couche 4), Network Load Balancer achemine le trafic vers les cibles au sein d'Amazon Virtual Private Cloud (Amazon VPC) et est capable de traiter des millions de requêtes par seconde tout en maintenant une latence très faible. Network Load Balancer est également optimisé pour gérer les pointes soudaines et les modèles de trafic changeants.

☐ **Équilibreur de charge classique (CLB)**  
Classic Load Balancer fournit un équilibrage de charge de base sur plusieurs instances Amazon EC2 et fonctionne à la fois au niveau de la demande et de connexion. Classic Load Balancer est destiné aux applications qui ont été construites dans le réseau EC2-Classique.

**Région**

EU (Paris) ▼

**Paramètres de service** [Informations](#)

Nombre d'équilibreurs de charge d'application

1

► Show calculations

## Unités de capacité de l'équilibreur de charge (LCU) [Informations](#)

Une LCU mesure les dimensions dans lesquelles l'équilibreur de charge d'application traite votre trafic (moyenne calculée sur une heure).

Veuillez spécifier au moins l'une des dimensions suivantes pour déterminer la tarification de la LCU.

**Octets traités (fonctions Lambda en tant que cibles)**  
Entrez le volume total de données traitées par équilibreur de charge d'application (ALB) des fonctions Lambda en tant que cibles.

Saisir la quantité GB per month ▼

**Octets traités (instances EC2 et d'adresses IP en tant que cibles)**  
Entrez le volume total de données traitées par ALB pour les instances EC2 et les adresses IP en tant que cibles.

Saisir la quantité GB per month ▼

**Nombre moyen de nouvelles connexions par Équilibreur de charge d'application (ALB)**

50 per second ▼

**Durée moyenne de la connexion**  
Entrez la durée moyenne pour chaque nouvelle connexion (si la durée est inférieure à 1 seconde, saisissez 1 seconde).

Saisir la quantité seconds ▼

**Nombre moyen de demandes par seconde par Équilibreur de charge d'application (ALB).**  
Entrez le nombre moyen de demandes par connexion.

Saisir la quantité

**Nombre moyen d'évaluations de règle par demande**

Saisir la quantité

- Le coût total pour la base de données seule est de 31.58\$ soit 28,50€.

Elastic Load Balancing estimate	
Application Load Balancer fixed hourly charges (monthly)	19.32 USD
Frais d'utilisation de l'équilibreur de charge d'application LCU (monthly)	12.26 USD
<b>Total monthly cost:</b>	<b>31.58 USD</b>

## CDN CloudFront

**Amazon CloudFront** est un réseau de diffusion mondial rapide de contenu. Dans la version gratuite de base il comprend 50Go de transfert sortant et 2 millions de requêtes HTTP/HTTPS. Il accélère la distribution des contenus des sites web statiques et dynamiques, tels que des fichiers .html, .css, .js, multimédias et image, aux utilisateurs.

Il est intégré aux services **Amazon EC2** et **Amazon ELB** et fonctionne de façon transparente.

## Stockage global

On utilisera **Amazon S3** (Simple Storage Service) pour stocker quelques fichiers de configuration qui seront utilisés de manière globale par l'application. Il servira à placer les données qui n'ont pas besoin d'être sécurisées comme les fichiers .html, .css... On n'a pas besoin d'une grande quantité de stockage donc son coût est négligeable dans notre étude.

## Système DNS Route 53

**Amazon Route 53** permet de faire le routage de façon optimale des utilisateurs vers notre application en garantissant la haute disponibilité. On peut gérer facilement l'acheminement des utilisateurs vers les microservices de notre application.

Il offre la possibilité de switcher sur une autre route à la volée si l'on a plusieurs serveurs en cas de panne ou de maintenance sur l'un d'eux.

## Récapitulatif.

Service Amazon	Quantité
Relational Database Service (RDS) for MySQL multi A-Z	1
Elastic Cloud (EC2)	6+3
Elastic Load Balancier (ELB)	1
CloudFront	1
Simple Storage Service (S3)	1
Route 53	1
Identity and Access Management (IAM)	1

On arrive à un total annuel de 2593,76\$ soit **2 340,93€** ou encore mensuellement 216.15\$ soit **195.07€**. D'autres possibilités existent comme **Microsoft Azure**, **Google Cloud** ou **Alibaba** qu'il serait possible de comparer dans une étude plus poussée de la solution technique qui n'est pas l'objet de ce document.

**Amazon RDS for MySQL**  
Région: EU (Paris)

Edit
Action ▼

**MySQL**

Type d'instance (db.t3.small), Quantité (1), Stockage pour chaque instance RDS (General Purpose), Quantité de stockage (30 Go)

Monthly: 47.03 USD

**Amazon EC2**  
Région: EU (Paris)

Edit
Action ▼

**Advance estimate (Estimation avancée)**

Système d'exploitation (linux), Instance EC2 avancée (t3a.small), Pricing strategy (1 Year None upfront), Stockage pour chaque instance EC2 (General Purpose), Quantité de stockage (30 Go), Fréquence d'instantané (59.83), Quantité modifiée par instantané (3 Go), Data Transfer, Coût du transfert de données (0)

Upfront: 0.00 USD  
Monthly: 137.31 USD



**Elastic Load Balancing**

Région: EU (Paris)

[Edit](#)[Action ▼](#)**Équilibreur de charge d'application**

Number of Application Load Balancers (1)

Monthly:

31.58 USD

**My Estimate**[Informations](#)[Ajouter un service](#)[Ajouter un groupe](#)[Action ▼](#)

First 12 months total

**2 593,76 USD**

Total upfront

**0,00 USD**

Total monthly

**216,15 USD**

## Table des matières.

Choix de l'architecture.....	2
Architecture Monolithique.....	2
Architecture microservices.....	3
Scalabilité.....	3
Maintenance.....	3
Encapsulation.....	3
Proxy.....	4
Décomposition de l'application.....	7
Microservice commande.....	7
Microservice stock.....	7
Microservice client.....	7
Microservice gestion.....	7
Microservice authentification.....	7
Description de l'architecture.....	8
Base de données.....	8
Application.....	10
Load balancer.....	13
CDN CloudFront.....	15
Récapitulatif.....	16
Table des matières.....	18