

# OC Pizza

## PizzaFlow

Dossier d'exploitation

Version 1.0

**Auteur**

Paul-Emmanuel DOS SANTOS FACAO  
*Analyste programmeur*

# TABLE DES MATIÈRES

<b>1 - Versions.....</b>	<b>7</b>
<b>2 - Introduction .....</b>	<b>8</b>
2.1 - Objet du document .....	8
2.2 - Références.....	8
2.3 - Diagramme de déploiement.....	9
<b>3 - Pré-requis .....</b>	<b>10</b>
3.1 - Création d'une paire de clé .....	10
3.2 - Système.....	12
3.2.1 - <i>Serveur de Base de données User</i> .....	12
3.2.1.1 - Description.....	12
3.2.1.2 - Caractéristiques techniques .....	12
3.2.2 - <i>Serveur de Base de données Stock</i> .....	13
3.2.2.1 - Description.....	13
3.2.2.2 - Caractéristiques techniques .....	14
3.2.3 - <i>Serveur de Base de données Gestion</i> .....	15
3.2.3.1 - Description.....	15
3.2.3.2 - Caractéristiques techniques .....	15
3.2.4 - <i>Serveur de configuration</i> .....	16
3.2.4.1 - Description.....	16
3.2.4.2 - Caractéristiques techniques .....	17
3.2.5 - <i>user-api</i> .....	18
3.2.5.1 - Description.....	18
3.2.5.2 - Caractéristiques techniques .....	18
3.2.6 - <i>stock-api</i> .....	19
3.2.6.1 - Description.....	19
3.2.6.2 - Caractéristiques techniques .....	20
3.2.7 - <i>web-api</i> .....	21
3.2.7.1 - Description.....	21
3.2.7.2 - Caractéristiques techniques .....	21
3.2.8 - <i>production-api</i> .....	22
3.2.8.1 - Description.....	22
3.2.8.2 - Caractéristiques techniques .....	23
3.2.9 - <i>gestion-api</i> .....	24
3.2.9.1 - Description.....	24
3.2.9.2 - Caractéristiques techniques .....	24
3.2.10 - <i>Serveur de Fichiers</i> .....	24
3.2.10.1 - Description.....	24
3.3 - Bases de données.....	25
3.4 - Les contrôleurs.....	26
3.5 - Web-services.....	26
3.6 - Serveur de configuration.....	26

3.7 - Les fichiers statiques .....	27
<b>4 - Procédure de déploiement .....</b>	<b>28</b>
4.1 - Configuration des outils .....	28
4.1.1 - Configuration de pgAdmin.....	28
4.1.2 - Configuration de PuTTY.....	31
4.2 - Installation de la base User.....	35
4.2.1 - Création du schéma de la base User .....	35
4.2.2 - Insérer les données initiales de la base User.....	37
4.3 - Installation de la base Stock.....	37
4.3.1 - Création du schéma de la base Stock.....	37
4.3.2 - Insérer les données initiales de la base Stock .....	37
4.4 - Installation de la base Gestion .....	37
4.4.1 - Création du schéma de la base Gestion .....	37
4.4.2 - Insérer les données initiales de la base Gestion.....	38
4.5 - Déploiement de config-server .....	38
4.5.1 - Récupération de l'archive .....	38
4.5.2 - Variables d'environnement.....	38
4.5.3 - Lancer le microservice.....	39
4.5.4 - Vérifications.....	39
4.6 - Déploiement de user-api.....	40
4.6.1 - Récupération de l'archive .....	40
4.6.2 - Variables d'environnement.....	40
4.6.3 - Configuration.....	41
4.6.4 - Lancer le microservice.....	42
4.6.5 - Vérifications.....	42
4.7 - Déploiement de stock-api.....	43
4.7.1 - Récupération de l'archive .....	43
4.7.2 - Variables d'environnement.....	43
4.7.3 - Configuration.....	44
4.7.4 - Lancer le microservice.....	45
4.7.5 - Vérifications.....	45
4.8 - Déploiement des fichiers statiques .....	46
4.8.1 - Récupération de l'archive .....	46
4.8.2 - Vérifications.....	47
4.9 - Déploiement de web-api.....	48
4.9.1 - Récupération de l'archive .....	48
4.9.2 - Variables d'environnement.....	48
4.9.3 - Configuration.....	49
4.9.4 - Lancer le microservice.....	50
4.9.5 - Vérifications.....	50
4.10 - Déploiement de production-api .....	51
4.10.1 - Récupération de l'archive .....	51
4.10.2 - Variables d'environnement.....	51
4.10.3 - Configuration .....	52

4.10.4 - Lancer le microservice .....	53
4.10.5 - Vérifications .....	53
4.11 - Déploiement de gestion-api.....	54
4.11.1 - Récupération de l'archive.....	54
4.11.2 - Variables d'environnement.....	54
4.11.3 - Configuration .....	55
4.11.4 - Lancer le microservice .....	56
4.11.5 - Vérifications .....	56
<b>5 - Procédure de démarrage / arrêt .....</b>	<b>58</b>
5.1 - Base de données User .....	58
5.1.1 - Préalable.....	58
5.1.2 - Démarrage.....	59
5.1.3 - Arrêt.....	60
5.2 - Base de données Stock .....	60
5.3 - Base de données Gestion.....	60
5.4 - Config-server .....	60
5.4.1 - Préalable.....	60
5.4.2 - Démarrage.....	61
5.4.3 - Arrêt.....	61
5.5 - user-api .....	61
5.5.1 - Préalable.....	61
5.5.2 - Démarrage.....	61
5.5.3 - Arrêt.....	61
5.6 - stock-api .....	62
5.6.1 - Préalable.....	62
5.6.2 - Démarrage.....	62
5.6.3 - Arrêt.....	62
5.7 - web-api .....	62
5.7.1 - Préalable.....	62
5.7.2 - Démarrage.....	63
5.7.3 - Arrêt.....	63
5.8 - production-api .....	63
5.8.1 - Préalable.....	63
5.8.2 - Démarrage.....	63
5.8.3 - Arrêt.....	64
5.9 - gestion-api.....	64
5.9.1 - Préalable.....	64
5.9.2 - Démarrage.....	64
5.9.3 - Arrêt.....	64
<b>6 - Procédure de mise à jour .....</b>	<b>65</b>
6.1 - Base de données User .....	65
6.1.1 - Préalable.....	65
6.1.2 - Mise à jour.....	65
6.1.3 - Finalisation .....	65

6.2 - Base de données Stock .....	65
6.2.1 - Préalable.....	65
6.2.2 - Mise à jour.....	66
6.2.3 - Finalisation .....	66
6.3 - Base de données Gestion.....	66
6.3.1 - Préalable.....	66
6.3.2 - Mise à jour.....	66
6.3.3 - Finalisation .....	67
6.4 - Microservice user-api.....	67
6.4.1 - Préalable.....	67
6.4.2 - Mise à jour.....	67
6.4.3 - Finalisation .....	68
6.5 - Microservice stock-api.....	68
6.5.1 - Préalable.....	68
6.5.2 - Mise à jour.....	68
6.5.3 - Finalisation .....	69
6.6 - Microservice web-api.....	69
6.6.1 - Préalable.....	69
6.6.2 - Mise à jour.....	69
6.6.3 - Finalisation .....	70
6.7 - Microservice production-api.....	70
6.7.1 - Préalable.....	70
6.7.2 - Mise à jour.....	70
6.7.3 - Finalisation .....	71
6.8 - Microservice gestion-api .....	71
6.8.1 - Préalable.....	71
6.8.2 - Mise à jour.....	71
6.8.3 - Finalisation .....	72
6.9 - Serveur de configuration.....	72
6.9.1 - Préalable.....	72
6.9.2 - Mise à jour.....	72
6.9.3 - Finalisation .....	73
<b>7 - Supervision/Monitoring .....</b>	<b>74</b>
7.1 - Supervision hardware des instances des microservices .....	74
7.2 - Supervision hardware des instances des bases de données .....	76
7.3 - CloudWatch Metrics .....	78
7.4 - Analyse des logs .....	79
7.5 - Les sondes.....	80
<b>8 - Procédure de sauvegarde et restauration .....</b>	<b>81</b>
8.1 - Base de données.....	81
8.1.1 - Sauvegarde d'une base de données .....	81
8.1.2 - Restauration d'une base de données.....	83
8.2 - Microservices .....	84
8.2.1 - Sauvegarde.....	84

8.2.1.1 - Sauvegarde sur AWS.....	84
8.2.1.2 - Backup. ....	86
8.2.2 - <i>Restauration ancienne configuration</i> .....	86
8.2.2.1 - Restauration des binaires .....	87
8.2.2.2 - Restauration d'une image d'instance sur AWS.....	87
<b>9 - Glossaire .....</b>	<b>91</b>

# 1 - VERSIONS

Auteur	Date	Description	Version
PEDSF	09/05/2020	Création du document	1.0
PEDSF	16/05/2020	Utilisation d'un serveur Nexus pour archiver les releases	2.0

## 2 - INTRODUCTION

### 2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application **PizzaFlow** à l'attention des mainteneurs et de l'équipe technique du client.

Les éléments du présent dossier découlent :

1. Des besoins exprimés par le client **OC Pizza** lors du premier contact,
2. De l'analyse des besoins de **OC Pizza**,
3. De la rédaction du dossier de conception fonctionnelle.
4. De la rédaction du dossier de conception technique.

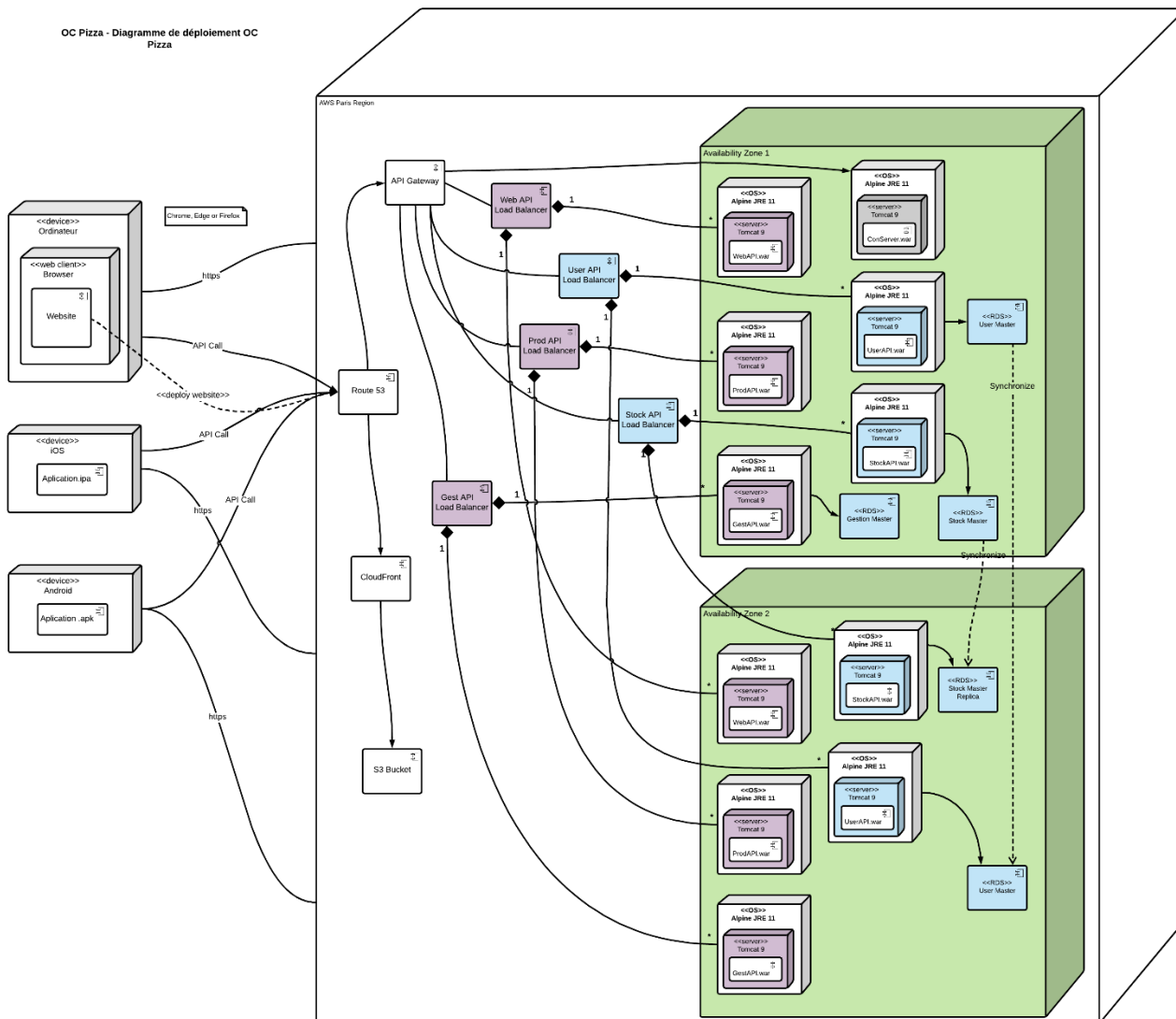
### 2.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants :

- **DCT - PDOCPizza\_01\_fonctionnelle** : Dossier de conception fonctionnelle de l'application.
- **DCT - PDOCPizza\_02\_technique** : Dossier de conception technique de l'application.



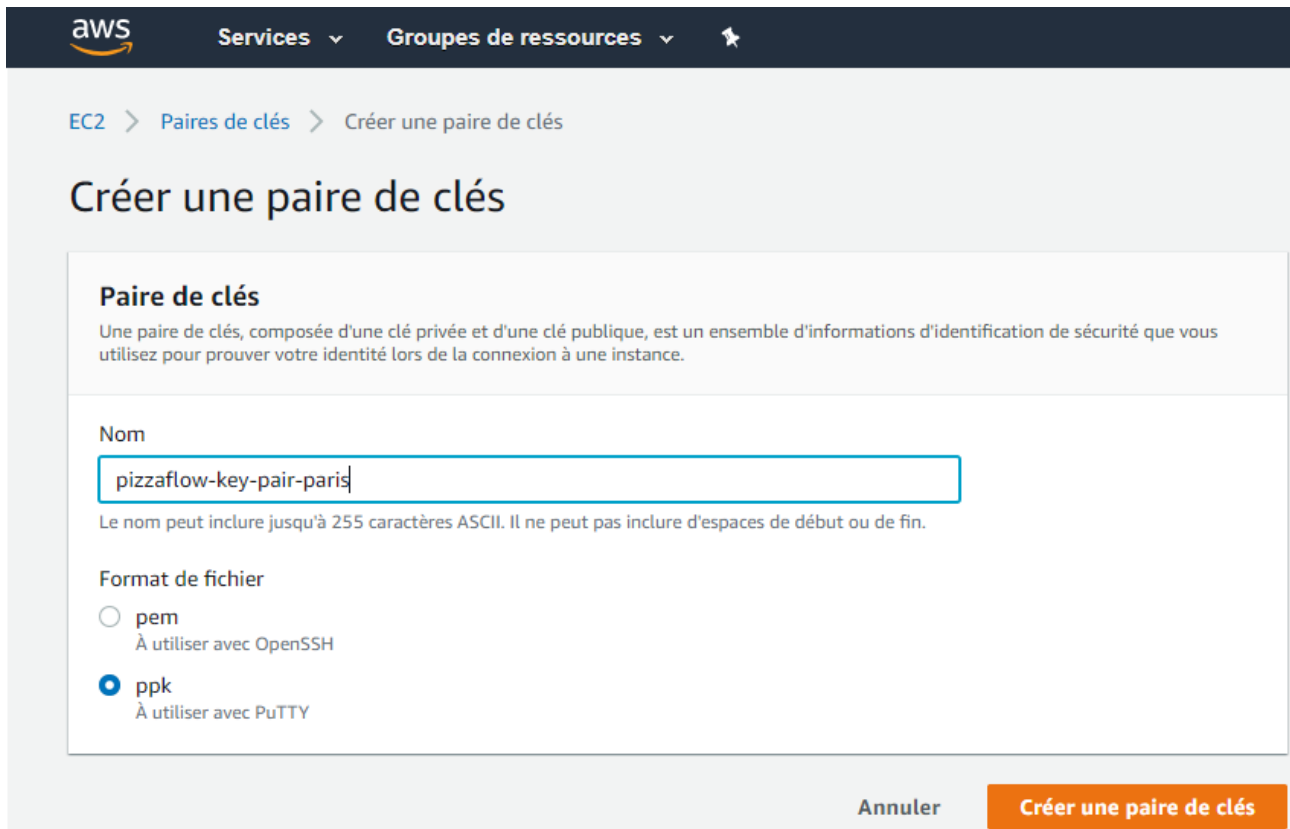
## 2.3 - Diagramme de déploiement



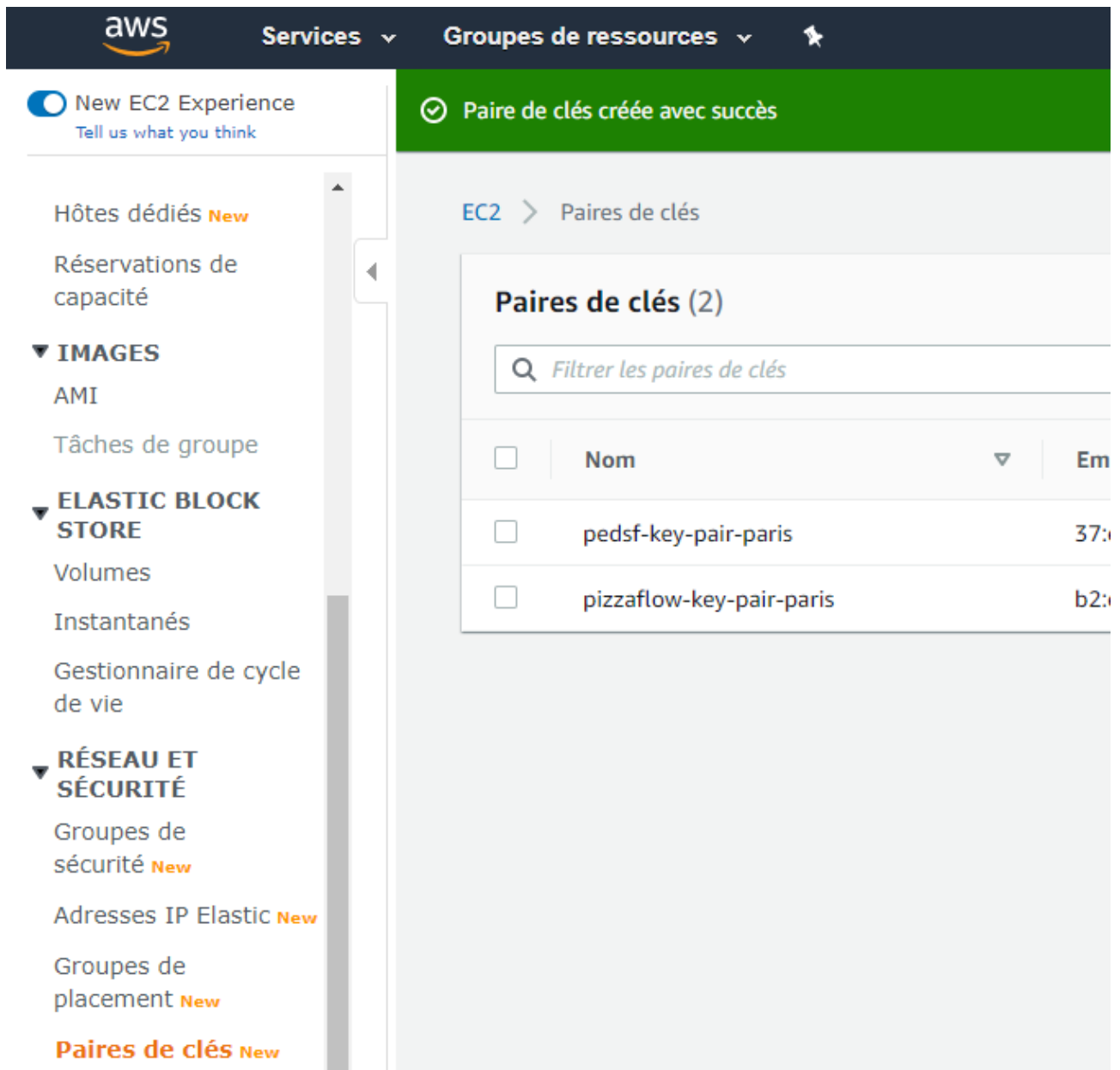
## 3 - PRÉ-REQUIS

### 3.1 - Création d'une paire de clé

Après création d'un compte sur AWS, on doit générer une paire de clés qui servira d'authentification pour accéder aux **VM**. Lors de la création des instances RDS et EC2, on spécifiera l'utilisation de cette clé. Dans la console de gestion des instances sur **AWS** et on sélectionne dans le menu de gauche "RESEAU ET SECURITE" le sous-menu "Paires de Clés" et on sélectionne créer une paire de clés. Le menu de création de paire de clé apparaît.



Rentrer le nom de la paire de clé et sélectionnez un format de fichier "ppk" pour l'utiliser avec **PuTTY**. Après validation sur la touche "Créer une paire de clés", la paire de clés générée apparaît dans l'écran des paires de clés.



**aws** Services ▾ Groupes de ressources ▾

☒ New EC2 Experience  
Tell us what you think

Hôtes dédiés **New**

Réervations de capacité

▼ **IMAGES**

AMI

Tâches de groupe

▼ **ELASTIC BLOCK STORE**

Volumes

Instantanés

Gestionnaire de cycle de vie

▼ **RÉSEAU ET SÉCURITÉ**

Groupes de sécurité **New**

Adresses IP Elastic **New**

Groupes de placement **New**

**Paires de clés** **New**

✓ Paire de clés créée avec succès

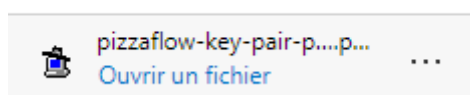
EC2 > Paires de clés

**Paires de clés (2)**

🔍 Filtrer les paires de clés

<input type="checkbox"/>	Nom	Em
<input type="checkbox"/>	pedsf-key-pair-paris	37:0
<input type="checkbox"/>	pizzaflow-key-pair-paris	b2:0

Le fichier PPK contenant la paire de clé générée est téléchargé via le navigateur. Copier le fichier sur votre ordinateur dans un endroit sécurisé.



## 3.2 - Système

### 3.2.1 - Serveur de Base de données User

#### 3.2.1.1 - Description

On utilise une instance **Amazon RDS (Relational Database Service)** for **PostgreSQL** avec une réservation d'une année pour réduire le coût et permettre de modifier en fonction de l'utilisation du site web le type de l'instance. On sélectionne l'option **Multi-AZ** pour avoir une seconde instance en standby synchronisée avec la première par sécurité au cas où un problème advienne sur la base de données master.

La base de données stockant uniquement les données des utilisateurs, elle n'a pas besoin d'une grande quantité de stockage. En outre, elle reçoit la majorité des requêtes sont lors de la connexion et sont seulement en lecture donc on n'a pas besoin de puissance de calcul.

#### 3.2.1.2 - Caractéristiques techniques

##### PostgreSQL instance specifications [Informations](#)

☐ Standard (Single-AZ)

Single-AZ deployment, Amazon RDS provisions a database in one Availability Zone.

☒ Multi-AZ

Multi-AZ deployment, Amazon RDS provisions and maintains a synchronous standby replica in a different Availability Zone.

##### Instance type

Q db.m5.large



##### db.m5.large

On-Demand hourly cost

0.412 USD

Reserved hourly cost (1YR, No upfront)

0.2855 USD

vCPUs

2

Memory

8 GiB

## Storage Informations

Enter the amount of storage you'd like for each instance.

Storage volume

General Purpose SSD (gp2)

Storage amount

5

GB per month

▼ Show calculations

5 GB per month x 0.266 USD x 1 instances = 1.33 USD (EBS Storage Cost)

**Storage pricing (monthly): 1.33 USD**

## 3.2.2 - Serveur de Base de données Stock

### 3.2.2.1 - Description

On utilise une instance **Amazon RDS for PostgreSQL** avec une réservation d'une année pour réduire le coût et permettre de modifier en fonction de l'utilisation du site web le type de l'instance. On sélectionne l'option **Multi-AZ** pour avoir une seconde instance en standby synchronisée avec la première par sécurité au cas où un problème advienne sur la base de données master.

La base de données stocke les informations des stocks, des paniers et des commandes. Elle a besoin de puissance de calcul et d'une réserve de stockage.

### 3.2.2.2 - Caractéristiques techniques

#### PostgreSQL instance specifications [Informations](#)

☐ **Standard (Single-AZ)**  
Single-AZ deployment, Amazon RDS provisions a database in one Availability Zone.

☒ **Multi-AZ**  
Multi-AZ deployment, Amazon RDS provisions and maintains a synchronous standby replica in a different Availability Zone.

#### Instance type

#### db.m5.2xlarge

On-Demand hourly cost  
1.648 USD

Reserved hourly cost (1YR, No upfront)  
1.1421 USD

vCPUs  
8

Memory  
32 GiB

#### Storage [Informations](#)

Enter the amount of storage you'd like for each instance.

#### Storage volume

#### Storage amount



#### ▼ Show calculations

30 GB per month x 0.266 USD x 1 instances = 7.98 USD (EBS Storage Cost)

**Storage pricing (monthly): 7.98 USD**

### 3.2.3 - Serveur de Base de données Gestion

#### 3.2.3.1 - Description

On utilise une instance **Amazon RDS** for **PostgreSQL** avec une réservation d'une année pour réduire le coût et permettre de modifier en fonction de l'utilisation du site web le type de l'instance. On sélectionne l'option **Multi-AZ** pour avoir une seconde instance en standby synchronisée avec la première par sécurité au cas où un problème advienne sur la base de données master.

La base de données stocke les données des ventes pour la gestion. Elle est utilisée par les Managers et la Direction donc elle a simplement besoin d'un espace de stockage conséquent sans puissance de calcul.

#### 3.2.3.2 - Caractéristiques techniques

##### PostgreSQL instance specifications [Informations](#)

☐ **Standard (Single-AZ)**  
Single-AZ deployment, Amazon RDS provisions a database in one Availability Zone.

☒ **Multi-AZ**  
Multi-AZ deployment, Amazon RDS provisions and maintains a synchronous standby replica in a different Availability Zone.

##### Instance type

Q db.m5.large

##### db.m5.large

On-Demand hourly cost  
0.412 USD

Reserved hourly cost (1YR, No upfront)  
0.2855 USD

vCPUs  
2

Memory  
8 GiB

## Storage Informations

Enter the amount of storage you'd like for each instance.

Storage volume

General Purpose SSD (gp2)

Storage amount

30

GB per month

▼ Show calculations

30 GB per month x 0.266 USD x 1 instances = 7.98 USD (EBS Storage Cost)

**Storage pricing (monthly): 7.98 USD**

## 3.2.4 - Serveur de configuration

### 3.2.4.1 - Description

Il sert juste à distribuer les configurations et ne nécessite pas de puissance de calcul. On prendra la plus petite des instances **Amazon EC2 (Elastic Cloud Compute)**.



### 3.2.4.2 - Caractéristiques techniques

#### Paramètre Informations

##### Système d'exploitation

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

Linux

##### Type d'instance

Effectuez une recherche sur le nom ou saisissez la condition de recherche de l'instance la moins chère qui répond à vos besoins.

☒ Saisir des exigences minimales pour chaque instance :

☐ Rechercher des instances en fonction du nom:

vCPU

1

Remove

Mémoire (Gio)

1

Remove

Add requirement

D'après vos entrées, voici l'instance EC2 la moins coûteuse :

##### t3a.micro

Coût horaire à la demande  
0.0106

vCPU  
2

GPU  
NA

Coût horaire standard réservé sur 1 an  
0.0067

Mémoire (Gio)  
1 GiB

Performances du réseau  
Low to Moderate

Le stockage sera partagé par toutes les instances Amazon EC2 pour mutualiser les ressources. On utilise 20Go de stockage avec 3 pics de 5Go modifié par jour vers les heures d'affluence et pour les transferts de données journaliers pour la maintenance en heure creuse.

## 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

20

Go

### Fréquence d'instantané

3 fois par jour

### Quantité modifiée par instantané

5

Go

## 3.2.5 - user-api

### 3.2.5.1 - Description

On prendra une petite instance avec 2 cœurs **Amazon EC2** scalable à 2 pendant 5h tous les jours.

### 3.2.5.2 - Caractéristiques techniques

#### EC2 Instances (131)

Selected Instance: **t3.micro**

🔍 Search by instance name or filter by keyword


2

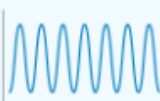
1 GiB

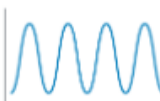
Any Network Perf...

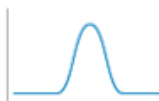
## Workload [Informations](#)

Sélectionnez le graphique qui représente le mieux votre charge de travail mensuelle.

☐ Utilisation constante  


☒ Pic de trafic quotidien  


☐ Pic de trafic hebdomadaire  


☐ Pic de trafic mensuel  


### ▼ Modèle de pic quotidien

[Supprimer le modèle](#)

#### Workload days

Select days for your workload pattern.

☒ Dimanche ☒ Lundi ☒ Mardi ☒ Mercredi ☒ Jeudi ☒ Vendredi ☒ Samedi

#### Baseline

Saisissez le nombre minimum d'instances dont vous avez besoin comme quantité pour votre charge de travail.

#### Peak

Saisissez le nombre maximal d'instances dont vous avez besoin au plus fort de votre charge de travail.

#### Durée du pic (heures, minutes)

Saisissez le nombre d'heures et de minutes pendant lesquelles le nombre maximal d'instances est exécuté.



## 3.2.6 - stock-api

### 3.2.6.1 - Description

On prendra une instance avec 2 cœurs **Amazon EC2** scalable à 3 pendant 5h tous les jours.

### 3.2.6.2 - Caractéristiques techniques

#### EC2 Instances (131)

Selected Instance: **t3.micro**

 Search by instance name or filter by keyword

2 ▼

1 GiB ▼

Any Network Perf... ▼

#### Workload [Informations](#)

Sélectionnez le graphique qui représente le mieux votre charge de travail mensuelle.

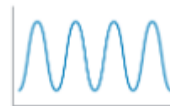
☐ Utilisation  
constante



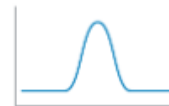
☒ Pic de trafic  
quotidien



☐ Pic de trafic  
hebdomadaire



☐ Pic de trafic  
mensuel



▼ **Modèle de pic quotidien**

**Supprimer le modèle**

#### Workload days

Select days for your workload pattern.

☒ Dimanche ☒ Lundi ☒ Mardi ☒ Mercredi ☒ Jeudi ☒ Vendredi ☒ Samedi

#### Baseline

Saisissez le nombre minimum d'instances dont vous avez besoin comme quantité pour votre charge de travail.

1

#### Peak

Saisissez le nombre maximal d'instances dont vous avez besoin au plus fort de votre charge de travail.

3

#### Durée du pic (heures, minutes)

Saisissez le nombre d'heures et de minutes pendant lesquelles le nombre maximal d'instances est exécuté.

5

0

### 3.2.7 - web-api

#### 3.2.7.1 - Description

On prendra une instance avec 4 cœurs **Amazon EC2** scalable à 3 pendant 5h tous les jours. On prend plus de puissance de calcul pour les microservices frontend.


#### 3.2.7.2 - Caractéristiques techniques

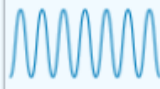
##### EC2 Instances (108)

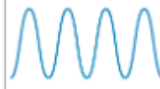
Selected Instance: c5.xlarge

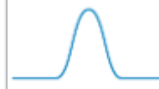
## Workload [Informations](#)

Sélectionnez le graphique qui représente le mieux votre charge de travail mensuelle.

☐ Utilisation constante  


☒ Pic de trafic quotidien  


☐ Pic de trafic hebdomadaire  


☐ Pic de trafic mensuel  


### ▼ Modèle de pic quotidien

[Supprimer le modèle](#)

#### Workload days

Select days for your workload pattern.

☒ Dimanche ☒ Lundi ☒ Mardi ☒ Mercredi ☒ Jeudi ☒ Vendredi ☒ Samedi

#### Baseline

Saisissez le nombre minimum d'instances dont vous avez besoin comme quantité pour votre charge de travail.

#### Peak

Saisissez le nombre maximal d'instances dont vous avez besoin au plus fort de votre charge de travail.

#### Durée du pic (heures, minutes)

Saisissez le nombre d'heures et de minutes pendant lesquelles le nombre maximal d'instances est exécuté.



## 3.2.8 - production-api

### 3.2.8.1 - Description

On prendra une instance avec 2 cœurs **Amazon EC2** scalable à 2 pendant 5h tous les jours. On prend plus de puissance de calcul pour les microservices frontend.

### 3.2.8.2 - Caractéristiques techniques

#### EC2 Instances (131)

Selected Instance: **t3.micro**

2 ▼

1 GiB ▼

Any Network Perf... ▼

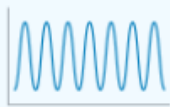
#### Workload [Informations](#)

Sélectionnez le graphique qui représente le mieux votre charge de travail mensuelle.

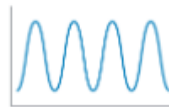
☐ Utilisation  
constante



☒ Pic de trafic  
quotidien



☐ Pic de trafic  
hebdomadaire



☐ Pic de trafic  
mensuel



▼ **Modèle de pic quotidien**

[Supprimer le modèle](#)

#### Workload days

Select days for your workload pattern.

☒ Dimanche ☒ Lundi ☒ Mardi ☒ Mercredi ☒ Jeudi ☒ Vendredi ☒ Samedi

#### Baseline

Saisissez le nombre minimum d'instances dont vous avez besoin comme quantité pour votre charge de travail.

1

#### Peak

Saisissez le nombre maximal d'instances dont vous avez besoin au plus fort de votre charge de travail.

2

#### Durée du pic (heures, minutes)

Saisissez le nombre d'heures et de minutes pendant lesquelles le nombre maximal d'instances est exécuté.

5

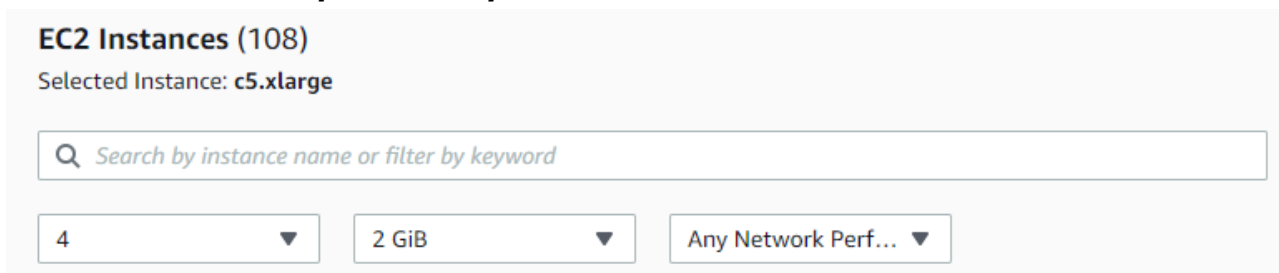
0

### 3.2.9 - gestion-api

#### 3.2.9.1 - Description

On prendra une instance avec 4 cœurs **Amazon EC2** sans load-balancing. On prend plus de puissance de calcul pour les microservices frontend de gestion.

#### 3.2.9.2 - Caractéristiques techniques



EC2 Instances (108)  
Selected Instance: c5.xlarge

Search by instance name or filter by keyword

4 ▼ 2 GiB ▼ Any Network Perf... ▼

### 3.2.10 - Serveur de Fichiers

#### 3.2.10.1 - Description

On utilise une instance **Amazon S3 (Simple Storage Service)** pour stocker les fichiers statiques des microservices frontend. On table sur 2 millions de transactions par mois et 20Go de données transférées.



## ▼ S3 Standard Informations

Les calculs ci-dessous excluent les remises de l'offre gratuite.

Stockage standard S3

Go per mois ▼

Demandes PUT, COPY, POST, LIST envoyées à S3 Standard

GET, SELECT et toutes les autres requêtes provenant de S3 Standard

Données renvoyées par S3 Select

Go per mois ▼

### 3.3 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour dans le repository **Nexus** privé.

Description	GroupId	ArtifactId	Version
<b>user schéma</b>	com.itcd.delivery	pizzaflow-user-init	1.0
<b>user datas</b>	com.itcd.delivery	pizzaflow-user-datas	1.0
<b>stock schéma</b>	com.itcd.delivery	pizzaflow-stock-init	1.0
<b>stock datas</b>	com.itcd.delivery	pizzaflow-stock-datas	1.0
<b>gestion schéma</b>	com.itcd.delivery	pizzaflow-gestion-init	1.0
<b>gestion datas</b>	com.itcd.delivery	pizzaflow-gestion-datas	1.0

### 3.4 - Les contrôleurs

Les contrôleurs suivants doivent être accessibles et à jour dans le repository **Nexus**.

Description	GroupId	ArtifactId	Version
<b>user-api</b>	com.itcd.delivery	pizzaflow-user-api	1.0
<b>stock-api</b>	com.itcd.delivery	pizzaflow-stock-api	1.0
<b>gestion-api</b>	com.itcd.delivery	pizzaflow-gestion-api	1.0

### 3.5 - Web-services

Les web services suivants doivent être accessibles et à jour dans le repository **Nexus**.

Description	GroupId	ArtifactId	Version
<b>web-api</b>	com.itcd.delivery	pizzaflow-web-api	1.0
<b>production-api</b>	com.itcd.delivery	pizzaflow-production-api	1.0
<b>gestion-api</b>	com.itcd.delivery	pizzaflow-gestion-api	1.0

### 3.6 - Serveur de configuration

Le serveur config-server et les fichiers de configurations doivent être accessibles et à jour dans le repository **Nexus**.

Description	GroupId	ArtifactId	Version
<b>fichier JAR</b>	com.itcd.delivery	config-server	1.0
<b>fichiers properties</b>	com.itcd.delivery	pizzaflow-prod-properties	1.0

### 3.7 - Les fichiers statiques

Les fichiers statiques pour les applications web doivent être accessibles et à jour dans le repository **Nexus**.

Description	GroupId	ArtifactId	Version
<b>images</b>	com.itcd.delivery	pizzaflow-prod-images	1.0
<b>templates</b>	com.itcd.delivery	pizzaflow-prod-templates	1.0
<b>locales</b>	com.itcd.delivery	pizzaflow-prod-locales	1.0
<b>css</b>	com.itcd.delivery	pizzaflow-prod-css	1.0
<b>js</b>	com.itcd.delivery	pizzaflow-prod-js	1.0

## 4 - PROCÉDURE DE DÉPLOIEMENT

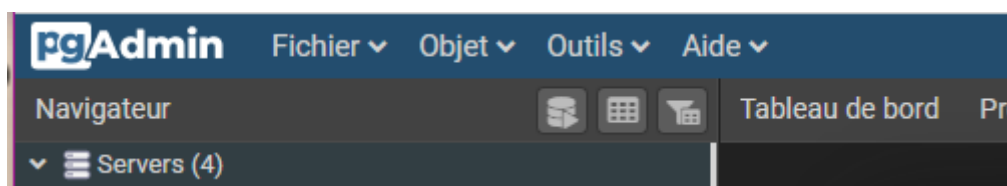
### 4.1 - Configuration des outils

#### 4.1.1 - Configuration de pgAdmin

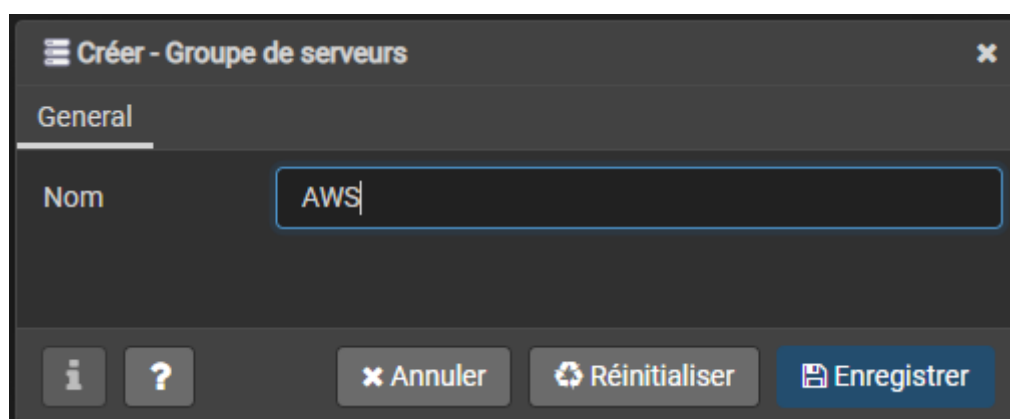
Pour accéder aux bases de données on doit installer **pgAdmin 4 (PostgreSQL Admin)**. Le lien pour installer le logiciel sur son ordinateur est :

<https://www.pgadmin.org/download/>

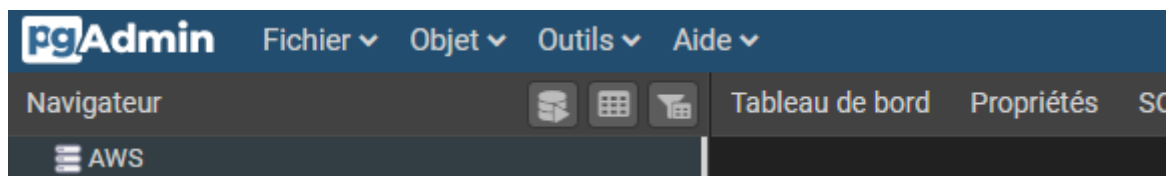
Une fois le logiciel installé, il faut configurer la connexion.



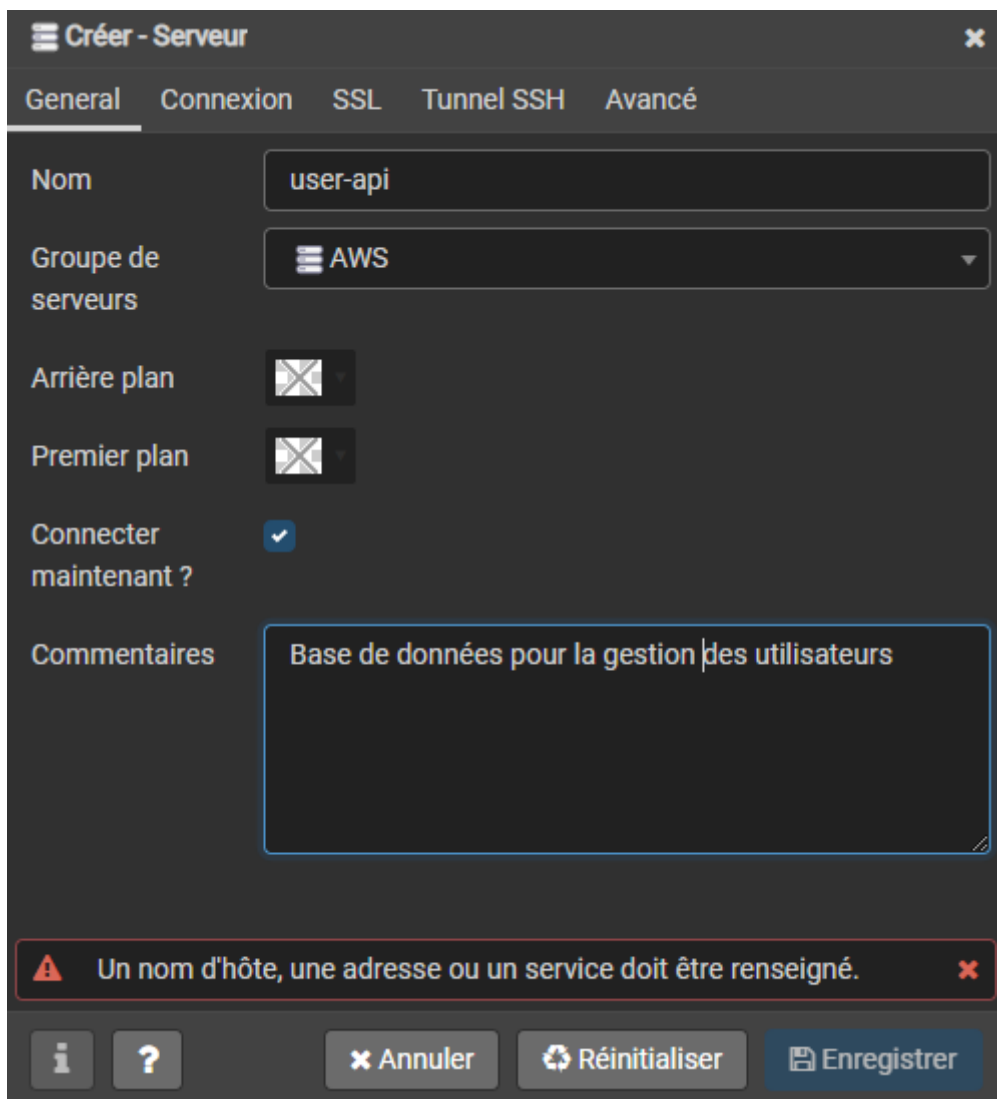
Il faut créer un groupe de serveur pour classer les connexions aux bases de données dans le menu "**Objet/Créer/Groupe de serveur**".



Sélectionner le groupe de serveur créé.

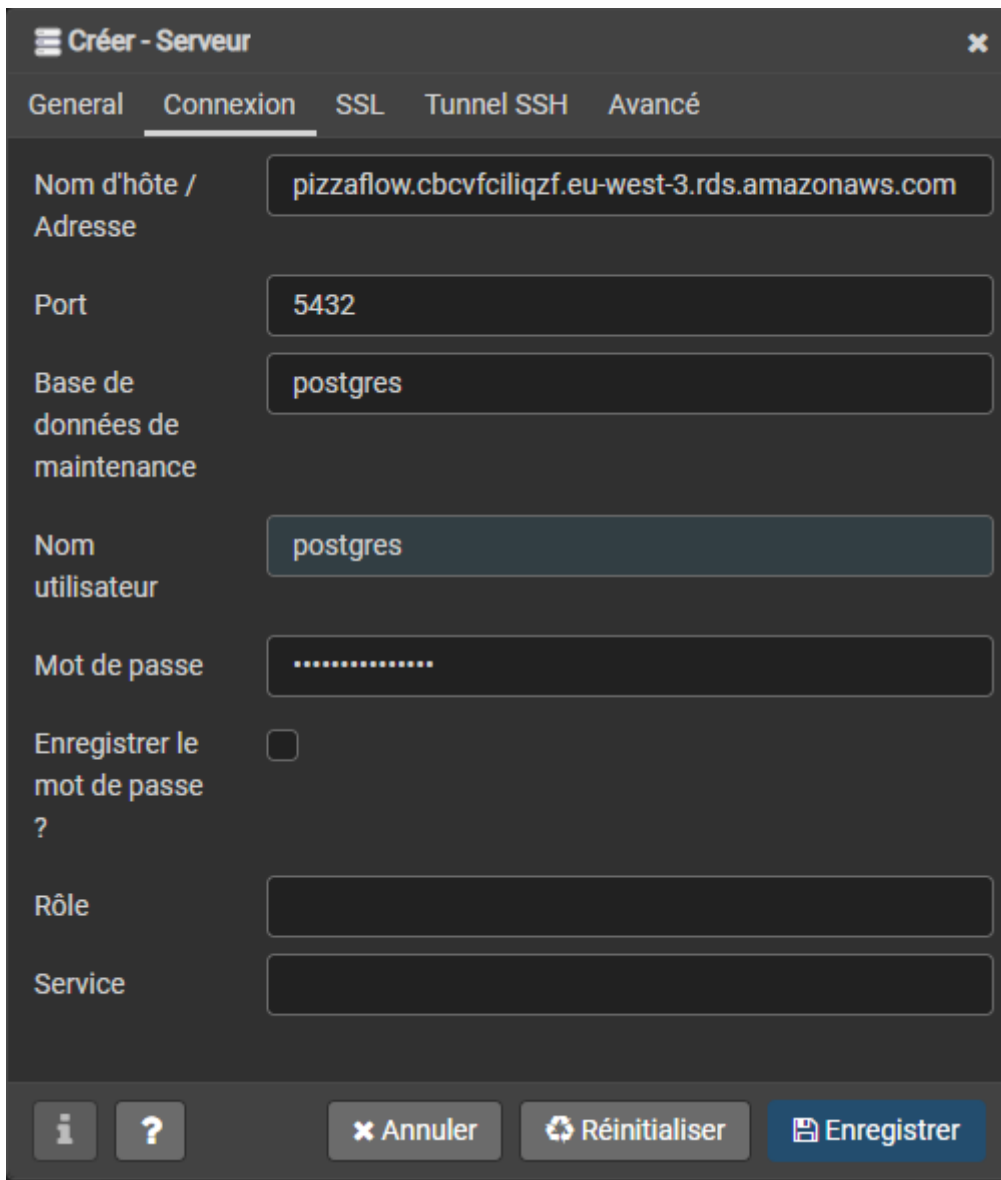


Cliquer sur le menu "**Objet/Créer/Serveur**" et indiquer le "**Nom**" de la connexion et une description dans la partie "Commentaires".



Sélectionner l'onglet "**Connexion**" pour renseigner les informations de connexion à la base de données. Entrer le "**Nom d'hôte/Adresse**" de la base de données, le "**Nom utilisateur**" et son "**Mot**

**de passe"** en fonction des indications données lors de la création de la base de données et enregistrer.



**Créer - Serveur**

General Connexion SSL Tunnel SSH Avancé

Nom d'hôte / Adresse: pizzaflow.cbvcfciliqzf.eu-west-3.rds.amazonaws.com

Port: 5432

Base de données de maintenance: postgres






Nom utilisateur: postgres

Mot de passe: .....

Enregistrer le mot de passe ? ☐

Rôle:

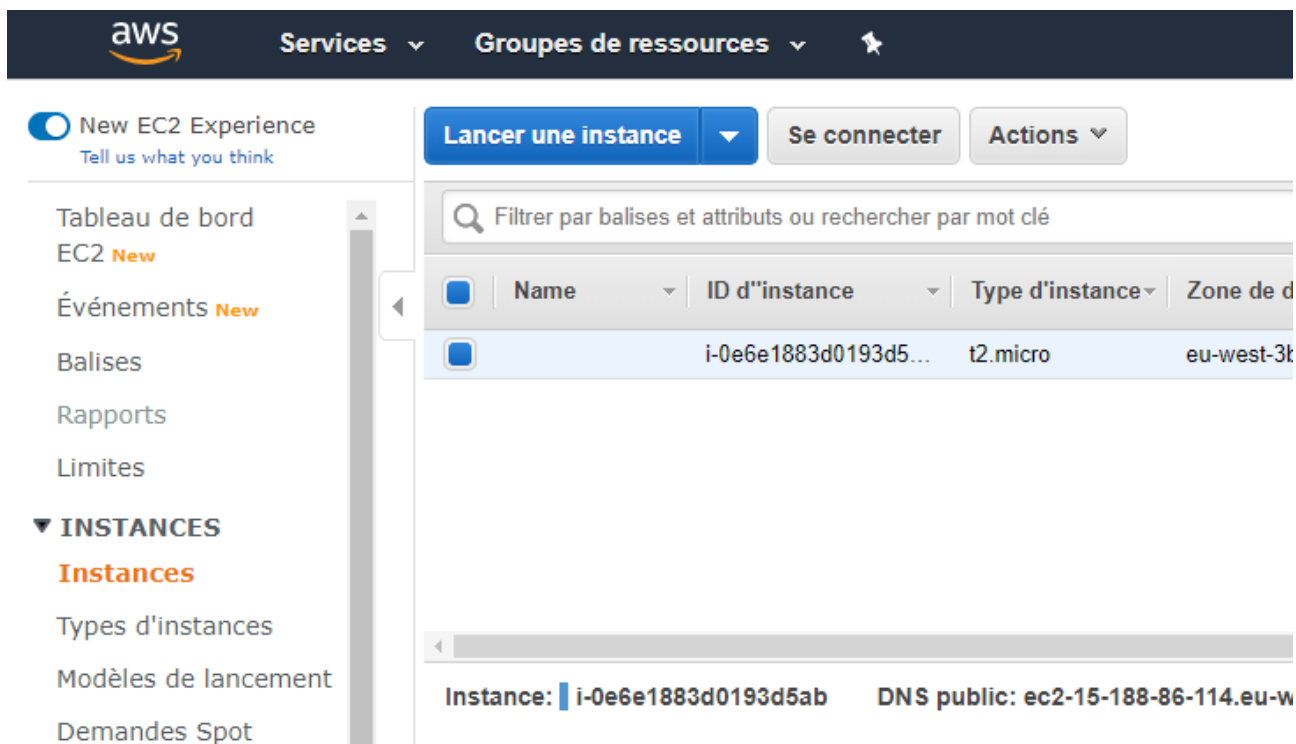
Service:

   Annuler  Réinitialiser  Enregistrer

Il faut créer une connexion pour chaque base de données (User, Stock et Gestion).

### 4.1.2 - Configuration de PuTTY

On doit créer une configuration pour chaque connexion. Pour avoir les informations nécessaires on se connecte sur la console de gestion des instances sur **AWS** et on sélectionne dans le menu de gauche "**INSTANCES**" le sous-menu "**Instances**".



The screenshot displays the AWS Management Console interface. At the top, the AWS logo and navigation tabs for 'Services' and 'Groupes de ressources' are visible. The left-hand navigation pane shows the 'New EC2 Experience' toggle and a list of menu items including 'Tableau de bord', 'EC2 New', 'Événements New', 'Balises', 'Rapports', 'Limites', and the expanded 'INSTANCES' section with 'Instances' highlighted. The main content area features a 'Lancer une instance' button, a 'Se connecter' button, and an 'Actions' dropdown. Below these is a search bar and a table of instances. The table has columns for 'Name', 'ID d'instance', 'Type d'instance', and 'Zone de d'. One instance is listed with ID 'i-0e6e1883d0193d5ab' and type 't2.micro'. At the bottom, the instance details are shown: 'Instance: i-0e6e1883d0193d5ab' and 'DNS public: ec2-15-188-86-114.eu-w'.

On sélectionne l'instance voulue puis on clique sur "**Se connecter**".

## Connectez-vous à votre instance



### Méthode de connexion

- ☒ Un client SSH autonome ⓘ  
☐ Session Manager ⓘ  
☐ Connexion d'instance EC2 (connexion SSH basée sur un navigateur) ⓘ

### Pour accéder à votre instance :

1. Ouvrez un client SSH. (découvrir comment [se connecter en utilisant PuTTY](#))
2. Recherchez votre fichier de clé privée (pizzaflow-key-pair-paris.pem). L'assistant détecte automatiquement la clé que vous avez utilisée pour lancer l'instance.
3. Votre clé ne doit pas être visible publiquement pour que SSH fonctionne. Utilisez cette commande, si nécessaire :

```
chmod 400 pizzaflow-key-pair-paris.pem
```

4. Connectez votre instance à l'aide de son DNS public :

```
ec2-15-236-51-101.eu-west-3.compute.amazonaws.com
```

### Exemple :

```
ssh -i "pizzaflow-key-pair-paris.pem" ec2-user@ec2-15-236-51-101.eu-west-3.compute.amazonaws.com
```

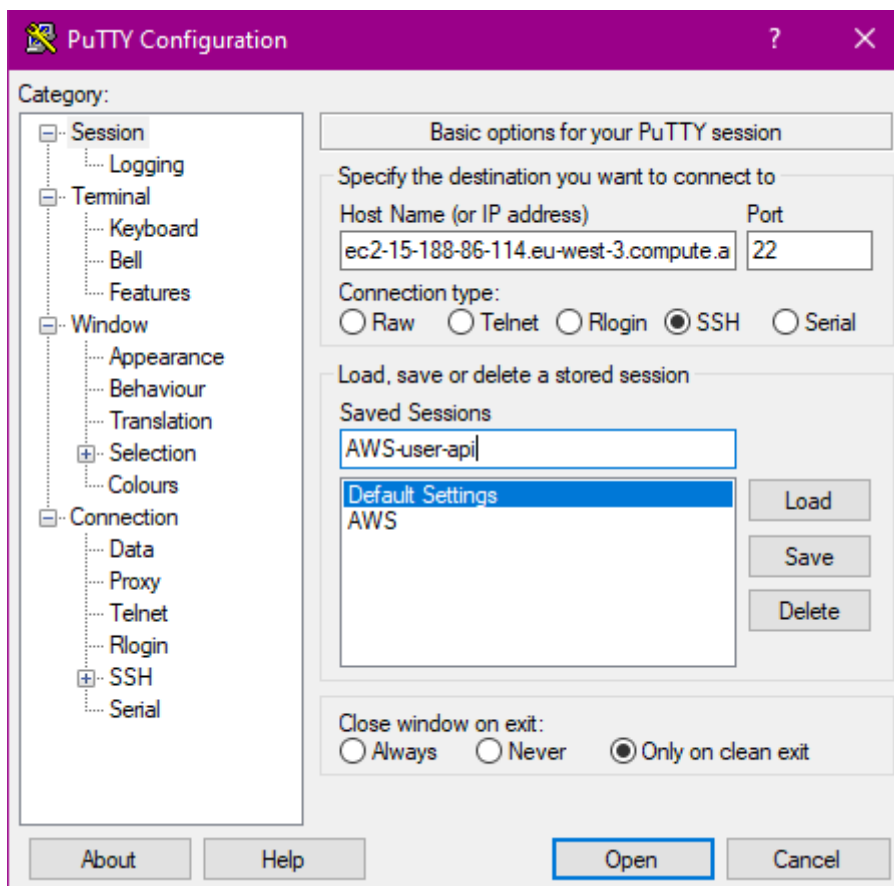
Veillez noter que, dans la plupart des cas, le nom d'utilisateur ci-dessus sera correct. Veillez cependant à lire les instructions d'utilisation de votre AMI afin de vous assurer que le propriétaire de l'AMI n'a pas changé le nom d'utilisateur par défaut de l'AMI.

Si vous avez besoin d'aide pour vous connecter à votre instance, veuillez consulter notre [documentation de connexion](#).

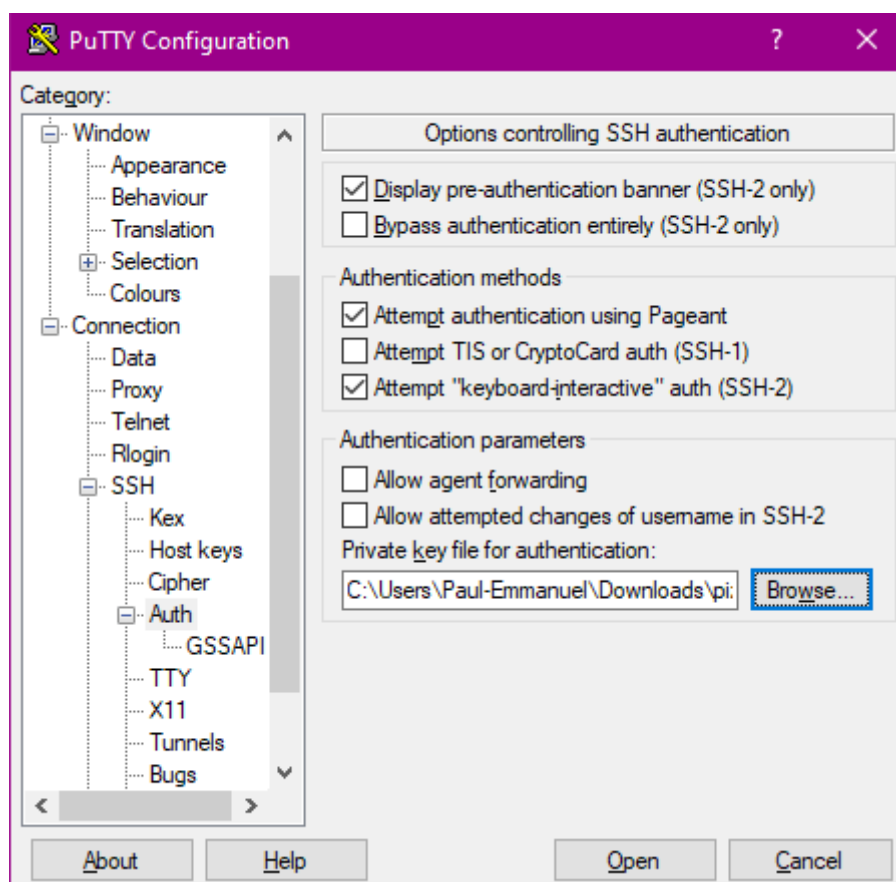
[Fermer](#)

- Copier le DNS public de l'instance et lancer PuTTY.
- Coller le nom du DNS dans la partie "**Host name**".
- Renseigner le "**Port**" 22.
- Sélectionner "**SSH**" dans le type de connexion.
- Renseigner le nom de sauvegarde de la connexion en spécifiant le nom du microservice dans "**Saved Session**".





- Dans le volet "**Catégorie**", entrer dans le sous-menu "**Connexion/SSH/Auth**".
- Choisir "**Parcourir**" et sélectionner le fichier de paire de clé **PPK**.



- Revenir sur le sous-menu **"Session"** et cliquer sur **"Save"** pour enregistrer les modifications.
- Cliquer sur **"Open"** pour ouvrir une session sur la machine distante.

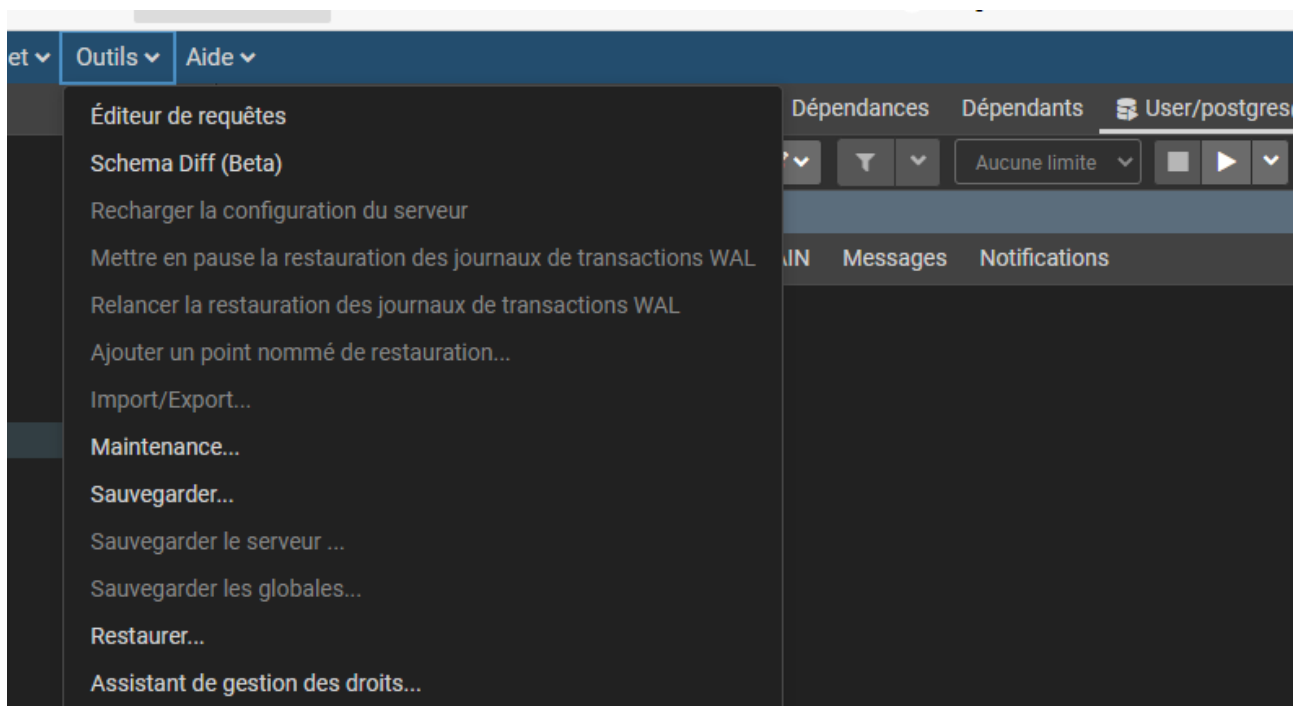
## 4.2 - Installation de la base User

### 4.2.1 - Création du schéma de la base User

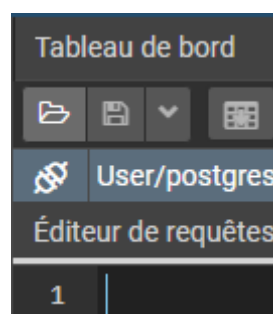
Télécharger le schéma de la base de données **User** dans le repository **Nexus** à l'adresse suivante :

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-user-schema-1.0>

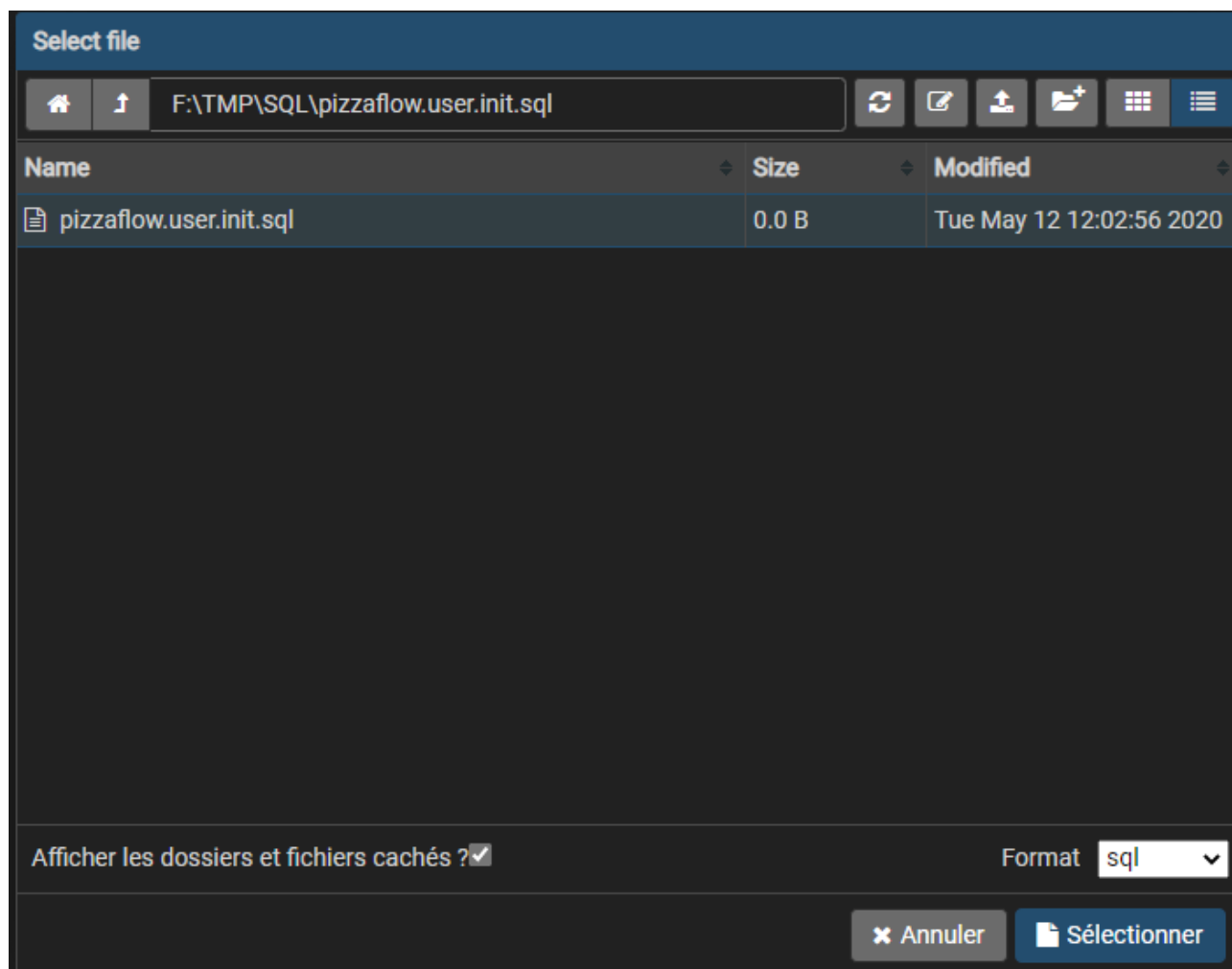
Dans **pgAdmin** il faut sélectionner la base de données **User** et le menu **Outils/Editeur de requêtes**. La fenêtre de l'éditeur s'ouvre dans la partie gauche de l'écran.



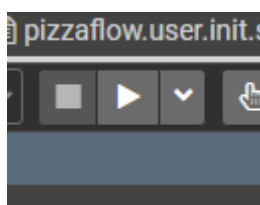
Cliquer sur l'icône "**Ouvrir fichier**" en haut à gauche de la fenêtre de requêtes.



Sélectionner le fichier **pizzaflow-user-schema-1.0.sql** et valider.



Dans la fenêtre des requêtes cliquer sur l'icône exécuter.



### 4.2.2 - Insérer les données initiales de la base User

Télécharger les données initiales de la base **User** dans le repository **Nexus** à l'adresse suivante :

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-user-datas-1.0>

Faire la même opération avec le fichier **pizzaflow-user-datas-1.0.sql** pour insérer les données dans la base.

## 4.3 - Installation de la base Stock

### 4.3.1 - Création du schéma de la base Stock

Récupérer le schéma de la base de données **Stock** sur le repository **Nexus** et effectuer les mêmes opérations que la base **User**.

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-stock-shema-1.0>

### 4.3.2 - Insérer les données initiales de la base Stock

Récupérer les données initiales de la base de données **Stock** sur le repository **Nexus** et effectuer les mêmes opérations que la base **User**.

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-stock-datas-1.0>

## 4.4 - Installation de la base Gestion

### 4.4.1 - Création du schéma de la base Gestion

Récupérer le schéma de la base de données **Gestion** sur le repository **Nexus** et effectuer les mêmes opérations que la base **User**.

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-gestion-schema-1.0>

#### 4.4.2 - Insérer les données initiales de la base Gestion

Récupérer les données initiales de la base de données **Gestion** sur le repository **Nexus** et effectuer les mêmes opérations que la base **User**.

<http://itcd.com/nexus/pizzaflow/com.itcd.delivery/1.0/pizzaflow-gestion-datas-1.0>

### 4.5 - Déploiement de config-server

#### 4.5.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **config-server** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O config-server.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a=config-server \
&v=1.0 \
&p=zip
/srv$ unzip config-server.zip -d config-server
/srv$ cd config-server
/srv/config-server$
```

#### 4.5.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/config-server/start.sh**.

Nom	Obligatoire	Description
<b>JASYPT_SECRET</b>	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
<b>PROPERTIES_PATH</b>	oui	Chemin du repository contenant les fichiers de propriétés

#### 4.5.3 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/config-server
/srv/config-server$ sudo chmod g+x u+x *.sh
/srv/config-server$ sudo ./start.sh
```

#### 4.5.4 - Vérifications

Exécuter la commande suivante pour vérifier que le EDGE microservice **config-server** fonctionne en tâche de fond.

```
/srv/config-server$ ps -l | grep config-server
4 S  0  1  0 0 80  0 - 967 -  pts/0  00:00:00 config-server
```

Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 4.6 - Déploiement de user-api

### 4.6.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **user-api** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O user-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a=user-api \
&v=1.0 \
&p=zip
/srv$ unzip user-api.zip -d user-api
/srv$ cd user-api
/srv/user-api$
```

### 4.6.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices **user-api** afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/user-api/start.sh**.

Nom	Obligatoire	Description
<b>JASYPT_SECRET</b>	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
<b>IT C. &amp; D.</b> www.itcd.com	IT Consulting & Development +33 1.23.45.67.89 – consulting@itcd.com S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A	



<b>CONFIG_IP</b>	oui	Adresse IP du serveur de configuration
<b>USER_DB_USERNAME</b>	oui	Identifiant de connexion à la base de données User
<b>USER_DB_PASSWORD</b>	oui	Mot de passe correspondant

### 4.6.3 - Configuration

Le fichier de configuration **user-api-prod.properties** comme les fichiers de configuration de tous les microservices sont dans le repository **git** suivant :

<https://github.com/pizzaflow/properties.git>

Lors du démarrage, le microservice le demande au serveur de configuration qui va le chercher et le transmet.

```
## port par défaut pour la première instance de user-api
## utiliser la gamme des ports 4XXX pour les duplications d'instances
server.port=4000

## Propriétés générales de l'application PizzaFlow
pizzaflow.nomPropriété=Valeur

## Propriétés particulières du microservice user-api
pizzaflow.user.nomPropriété=Valeur

## configuration du pool de connexion par défaut
spring.datasource.hikari.connectionTimeout=20000
spring.datasource.hikari.maximumPoolSize=5

## configuration de PostgreSQL remplacer localhost par la vraie adresse IP
spring.datasource.url=jdbc:postgresql://localhost:5432/user
spring.datasource.driverClassName=org.postgresql.Driver

## identifiants temporaire de connexion à changer et crypter avec Jasypt pour la production
spring.datasource.username=${USER_DB_USERNAME}
spring.datasource.password=${USER_DB_PASSWORD}

# DANGER!! mettre à create pour refaire le schéma
spring.jpa.hibernate.ddl-auto=update

## Propriétés pour les logs
logging.level.org.springframework.web=ERROR
```

```
logging.level.com.pizzaflow=DEBUG

## le pattern pour la console
logging.pattern.console= "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"

## le pattern pour le nom du fichier
logging.pattern.file= "%d{yyyy-MM-dd HH:mm:ss}[%thread]%-5level %logger{36} - %msg%n"

## le nom du fichier de log
logging.file=/var/log/pizzaflow/user-api.log
```

#### 4.6.4 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice **user-api** en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/user-api
/srv/user-api$ sudo chmod g+x u+x *.sh
/srv/user-api$ sudo ./start.sh
```

#### 4.6.5 - Vérifications

Exécuter la commande suivante pour vérifier que le microservice **user-api** fonctionne en tâche de fond.

```
/srv/user-api$ ps -l | grep user-api
4 S   0   1   0 0 80  0 - 967 -   pts/0   00:00:00 user-api
```

Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 4.7 - Déploiement de stock-api

### 4.7.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **stock-api** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O stock-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a= stock-api \
&v=1.0 \
&p=zip
/srv$unzip stock-api.zip -d stock-api
/srv$cd stock-api
/srv/stock-api$
```

### 4.7.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/stock-api/start.sh**.

Nom	Obligatoire	Description
JASYPT_SECRET	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
IT C. & D. www.itcd.com	IT Consulting & Development +33 1.23.45.67.89 – consulting@itcd.com S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A	

<b>CONFIG_IP</b>	oui	Adresse IP du serveur de configuration
<b>STOCK_DB_USERNAME</b>	oui	Identifiant de connexion à la base de données Stock
<b>STOCK_DB_PASSWORD</b>	oui	Mot de passe correspondant

### 4.7.3 - Configuration

Le fichier de configuration **stock-api-prod.properties** comme les fichiers de configuration de tous les microservices sont dans le repository **git** suivant :

<https://github.com/pizzaflow/properties.git>

Lors du démarrage, le microservice le demande au serveur de configuration qui va le chercher et le transmet.

```
## port par défaut pour la première instance de stock-api
## utiliser la gamme des ports 5XXX pour les duplications d'instances
server.port=5000

## Propriétés générales de l'application PizzaFlow
pizzaflow.nomPropriété=Valeur

## Propriétés particulières du microservice stock-api
pizzaflow.stock.nomPropriété=Valeur

## configuration du pool de connexion par défaut
spring.datasource.hikari.connectionTimeout=20000
spring.datasource.hikari.maximumPoolSize=5

## configuration de PostgreSQL remplacer localhost par la vraie adresse IP
spring.datasource.url=jdbc:postgresql://localhost:5432/stock
spring.datasource.driverClassName=org.postgresql.Driver

## identifiants temporaire de connexion à changer et crypter avec Jasypt pour la production
spring.datasource.username=${STOCK_DB_USERNAME}
spring.datasource.password=${STOCK_DB_PASSWORD}

# DANGER!! mettre à create pour refaire le schéma
spring.jpa.hibernate.ddl-auto=update

## Properties pour les logs
```

```
logging.level.org.springframework.web=ERROR
logging.level.com.pizzaflow=DEBUG

## le pattern pour la console
logging.pattern.console= "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"

## le pattern pour le nom du fichier
logging.pattern.file= "%d{yyyy-MM-dd HH:mm:ss}[%thread]%-5level %logger{36} - %msg%n"

## le nom du fichier de log
logging.file=/var/log/pizzaflow/stock-api.log
```

#### 4.7.4 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/stock-api
/srv/stock-api$ sudo chmod g+x u+x *.sh
/srv/stock-api$ sudo ./start.sh
```

#### 4.7.5 - Vérifications

Exécuter la commande suivante pour vérifier que le microservice fonctionne en tâche de fond.

```
/srv/stock-api$ ps -l | grep stock-api
4 S  0  1  0 0 80  0 - 967 - pts/0  00:00:00 stock-api
```

Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 4.8 - Déploiement des fichiers statiques

### 4.8.1 - Récupération de l'archive

Se connecter au bucket **S3** avec **PuTTY** et créer un répertoire **/resources** et se placer dedans.

```
$mkdir /resources  
$cd /resources
```

Récupérer le **ZIP** des fichiers properties et décompresser-le qui est dans un repository **Nexus** avec la commande :

```
/resources$ wget -O images.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a= pizzaflow-prod-images \
&v=1.0 \
&p=zip  
/resources$ unzip images.zip -d images  
/resources$ wget -O templates.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
\
r=pizzaflow \
&g=com.itcd.delivery \
&a= pizzaflow-prod-templates \
&v=1.0 \
&p=zip  
/resources$ unzip templates.zip -d templates  
  
/resources$ wget -O locales.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a= pizzaflow-prod-locales \
&v=1.0 \
&p=zip  
/resources$ unzip locales.zip -d locales  
  
/resources$ wget -O css.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
```

```
&a= pizzaflow-prod-css \  
&v=1.0 \  
&p=zip  
/resources$unzip css.zip -d css  
  
/resources$ wget -O js.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= pizzaflow-prod-js \  
&v=1.0 \  
&p=zip  
/resources$unzip js.zip -d js
```

### 4.8.2 - Vérifications

Exécuter la commande suivante pour vérifier que les répertoires ont bien été créés.

```
/resources$ ls -l  
total 92  
-rw-r--r-- 1 root root 125 May 16 08:21 css.zip  
drwxr-xr-x 1 root root 1204 May 16 08:21 css  
-rw-r--r-- 1 root root 29492 May 16 08:21 images.zip  
drwxr-xr-x 1 root root 4096 May 16 08:21 images  
-rw-r--r-- 1 root root 7621 May 16 08:21 js.zip  
drwxr-xr-x 1 root root 4096 May 16 08:21 js  
-rw-r--r-- 1 root root 1245 May 16 08:21 locales.zip  
drwxr-xr-x 1 root root 4096 May 16 08:21 locales  
-rw-r--r-- 1 root root 7621 May 16 08:21 template.zip  
drwxr-xr-x 1 root root 4096 May 16 08:21 template
```

On voit les archives récupérées et un répertoire pour chaque archive.

## 4.9 - Déploiement de web-api

### 4.9.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **web-api** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O web-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a=web-api \
&v=1.0 \
&p=zip
/srv$ unzip web-api.zip -d web-api
/srv$ cd web-api
/srv/web-api$
```

### 4.9.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices **web-api** afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/web-api/start.sh**.

Nom	Obligatoire	Description
<b>JASYPT_SECRET</b>	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
<b>IT C. &amp; D.</b> www.itcd.com	IT Consulting & Development +33 1.23.45.67.89 – consulting@itcd.com S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A	



<b>CONFIG_IP</b>	oui	Adresse IP du serveur de configuration
<b>USER_API_IP</b>	oui	Adresse IP de <b>user-api</b>
<b>STOCK_API_IP</b>	oui	Adresse IP de <b>stock-api</b>
<b>RESOURCES_PATH</b>	oui	Chemin vers le bucket des ressources

### 4.9.3 - Configuration

Le fichier de configuration **web-api-prod.properties** comme les fichiers de configuration de tous les microservices sont dans le repository **git** suivant :

<https://github.com/pizzaflow/properties.git>

Lors du démarrage, le microservice le demande au serveur de configuration qui va le chercher et le transmet.

```
## port par défaut pour la première instance de web-api
## utiliser la gamme des ports 7XXX pour les duplications d'instances
server.port=7000

## Propriétés générales de l'application PizzaFlow
pizzaflow.nomPropriété=Valeur

## Propriétés particulières du microservice web-api
pizzaflow.web.nomPropriété=Valeur

## Properties pour les logs
logging.level.org.springframework.web=ERROR
logging.level.com.pizzaflow=DEBUG

## le pattern pour la console
logging.pattern.console= "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"

## le pattern pour le nom du fichier
logging.pattern.file= "%d{yyyy-MM-dd HH:mm:ss}[%thread]%-5level %logger{36} - %msg%n"

## le nom du fichier de log
logging.file=/var/log/pizzaflow/web-api.log
```

#### 4.9.4 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice **web-api** en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/web-api
/srv/web-api$ sudo chmod g+x u+x *.sh
/srv/web-api$ sudo ./start.sh
```

#### 4.9.5 - Vérifications

Exécuter la commande suivante pour vérifier que le microservice **web-api** fonctionne en tâche de fond.

```
/srv/web-api$ ps -l | grep web-api
4 S  0  1  0 0 80  0 - 967 - pts/0  00:00:00 web-api
```

Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 4.10 - Déploiement de production-api

### 4.10.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **production-api** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O production-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a= production-api \
&v=1.0 \
&p=zip
/srt$unzip production-api.zip -d production-api
/srt$cd production-api
/srt/ production-api$
```

### 4.10.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/production-api/start.sh**.

Nom	Obligatoire	Description
<b>JASYPT_SECRET</b>	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
<b>IT C. &amp; D.</b> www.itcd.com	IT Consulting & Development +33 1.23.45.67.89 – consulting@itcd.com S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A	

<b>CONFIG_IP</b>	oui	Adresse IP du serveur de configuration
<b>USER_API_IP</b>	oui	Adresse IP de <b>user-api</b>
<b>STOCK_API_IP</b>	oui	Adresse IP de <b>stock-api</b>
<b>WEB_API_IP</b>	oui	Adresse IP de <b>web-api</b>
<b>RESOURCES_PATH</b>	oui	Chemin vers le bucket des ressources

### 4.10.3 - Configuration

Le fichier de configuration **production-api-prod.properties** comme les fichiers de configuration de tous les microservices sont dans le repository **git** suivant :

<https://github.com/pizzaflow/properties.git>

Lors du démarrage, le microservice le demande au serveur de configuration qui va le chercher et le transmet.

```
## port par défaut pour la première instance de production-api
## utiliser la gamme des ports 8XXX pour les duplications d'instances
server.port=8000

## Propriétés générales de l'application PizzaFlow
pizzaflow.nomPropriété=Valeur

## Propriétés particulières du microservice production-api
pizzaflow.production.nomPropriété=Valeur

## Properties pour les logs
logging.level.org.springframework.web=ERROR
logging.level.com.pizzaflow=DEBUG

## le pattern pour la console
logging.pattern.console= "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"

## le pattern pour le nom du fichier
logging.pattern.file= "%d{yyyy-MM-dd HH:mm:ss}[%thread]%-5level %logger{36} - %msg%n"

## le nom du fichier de log
logging.file=/var/log/pizzaflow/production-api.log
```

#### 4.10.4 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/production-api  
/srv/ production-api$ sudo chmod g+x u+x *.sh  
/srv/ production-api$ sudo ./start.sh
```

#### 4.10.5 - Vérifications

Exécuter la commande suivante pour vérifier que le microservice fonctionne en tâche de fond.

```
/srv/ production-api$ ps -l | grep production-api  
4 S  0  1  0 0 80  0 - 967 -  pts/0  00:00:00 production-api
```

Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 4.11 - Déploiement de gestion-api

### 4.11.1 - Récupération de l'archive

Se connecter à la **VM** du microservice avec **PuTTY** et se placer dans le répertoire **/srv** de l'instance.

```
$cd /srv
```

Récupérer le **ZIP** de **gestion-api** qui est dans un repository **Nexus** et décompresser-le avec les commandes :

```
/srv$ wget -O gestion-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect? \
r=pizzaflow \
&g=com.itcd.delivery \
&a= gestion-api \
&v=1.0 \
&p=zip
/srt$unzip gestion-api.zip -d gestion-api
/srt$cd gestion-api
/srt/ gestion-api$
```

### 4.11.2 - Variables d'environnement

On doit spécifier les variables d'environnement pour les microservices afin qu'ils puissent récupérer leur configuration. On définit une variable d'environnement avec la commande :

```
$export NOM_VARIABLE=valeur
```

Ou peut renseigner ces valeurs dans le script **/srv/gestion-api/start.sh**.

Nom	Obligatoire	Description
JASYPT_SECRET	oui	Clé pour de cryptage des données sensibles des fichiers de propriétés.
CONFIG_IP	oui	Adresse IP du serveur de configuration
USER_API_IP	oui	Adresse IP de <b>user-api</b>
STOCK_API_IP	oui	Adresse IP de <b>stock-api</b>
WEB_API_IP	oui	Adresse IP de <b>web-api</b>
RESOURCES_PATH	oui	Chemin vers le bucket des ressources
GESTION_DB_USERNAME	oui	Identifiant de connexion à la base de données Gestion
GESTION_DB_PASSWORD	oui	Mot de passe correspondant

### 4.11.3 - Configuration

Le fichier de configuration **gestion-api-prod.properties** comme les fichiers de configuration de tous les microservices sont dans le repository **git** suivant :

<https://github.com/pizzaflow/properties.git>

Lors du démarrage, le microservice le demande au serveur de configuration qui va le chercher et le transmet.

```
## port par défaut pour la première instance de production-api
## utiliser la gamme des ports 9XXX pour les duplications d'instances
server.port=9000

## Propriétés générales de l'application PizzaFlow
pizzaflow.nomPropriété=Valeur

## Propriétés particulières du microservice gestion-api
pizzaflow.gestion.nomPropriété=Valeur

## configuration du pool de connexion par défaut
spring.datasource.hikari.connectionTimeout=20000
spring.datasource.hikari.maximumPoolSize=5

## configuration de PostgreSQL remplacer localhost par la vraie adresse IP
spring.datasource.url=jdbc:postgresql://localhost:5432/stock
spring.datasource.driverClassName=org.postgresql.Driver
```

```
## identifiants temporaire de connexion à changer et crypter avec Jasypt pour la production
spring.datasource.username=${GESTION_DB_USERNAME}
spring.datasource.password=${GESTION_DB_PASSWORD}

# DANGER!! mettre à create pour refaire le schéma
spring.jpa.hibernate.ddl-auto=update

## Properties pour les logs
logging.level.org.springframework.web=ERROR
logging.level.com.pizzaflow=DEBUG

## le pattern pour la console
logging.pattern.console= "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"

## le pattern pour le nom du fichier
logging.pattern.file= "%d{yyyy-MM-dd HH:mm:ss}[%thread]%-5level %logger{36} - %msg%n"

## le nom du fichier de log
logging.file=/var/log/pizzaflow/gestion-api.log
```

#### 4.11.4 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice en tâche de fond après avoir défini les variables d'environnement et déployer les fichiers statiques. Se placer dans le répertoire du microservice, rendre exécutable les scripts et exécuter le script de démarrage.

```
$ cd /srv/gestion-api
/srv/ gestion-api$ sudo chmod g+x u+x *.sh
/srv/ gestion-api$ sudo ./start.sh
```

#### 4.11.5 - Vérifications

Exécuter la commande suivante pour vérifier que le microservice fonctionne en tâche de fond.

```
/srv/ gestion-api$ ps -l | grep gestion-api
```

IT C. & D.

www.itcd.com

IT Consulting & Development +33 1.23.45.67.89 – consulting@itcd.com

S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A



4 S	0	1	0 0 80	0 - 967 -	pts/0	00:00:00	gestion-api
-----	---	---	--------	-----------	-------	----------	-------------

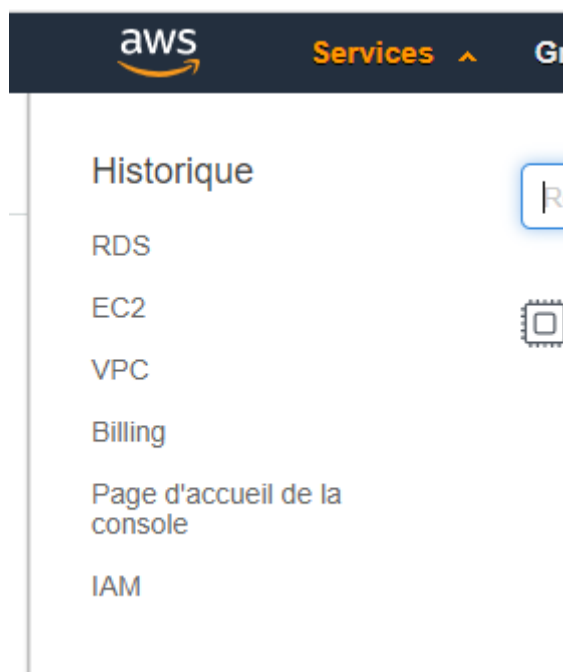
Une ligne est affichée donc le microservice fonctionne et l'identifiant du process est indiqué en 4<sup>ème</sup> colonne. Se connecter sur **logit.io** pour avoir une confirmation du fonctionnement en regardant les logs.

## 5 - PROCÉDURE DE DÉMARRAGE / ARRÊT

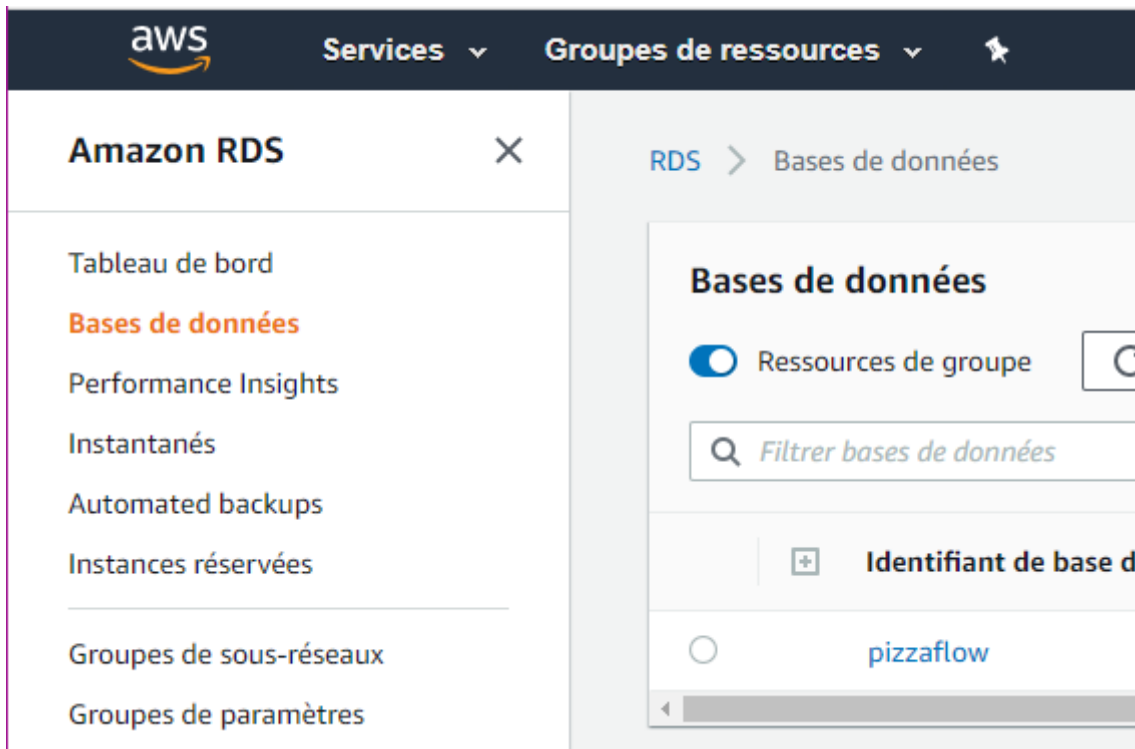
### 5.1 - Base de données User

#### 5.1.1 - Préalable

Se connecter à l'interface de gestion de **AWS** et sélectionner le service **RDS**.



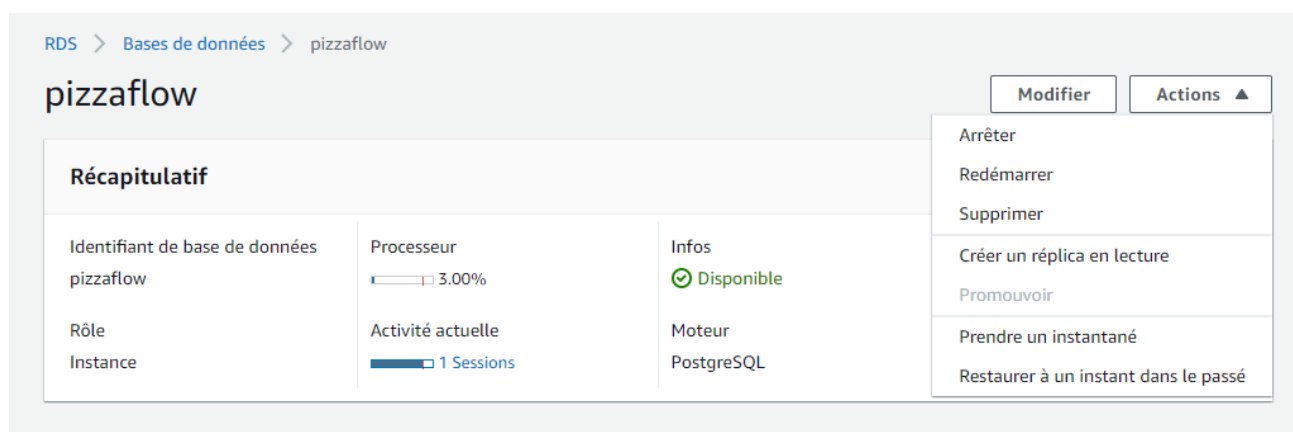
Choisir "**Bases de données**" pour voir les instances des bases de données.



Cliquer sur la base de données **User**.

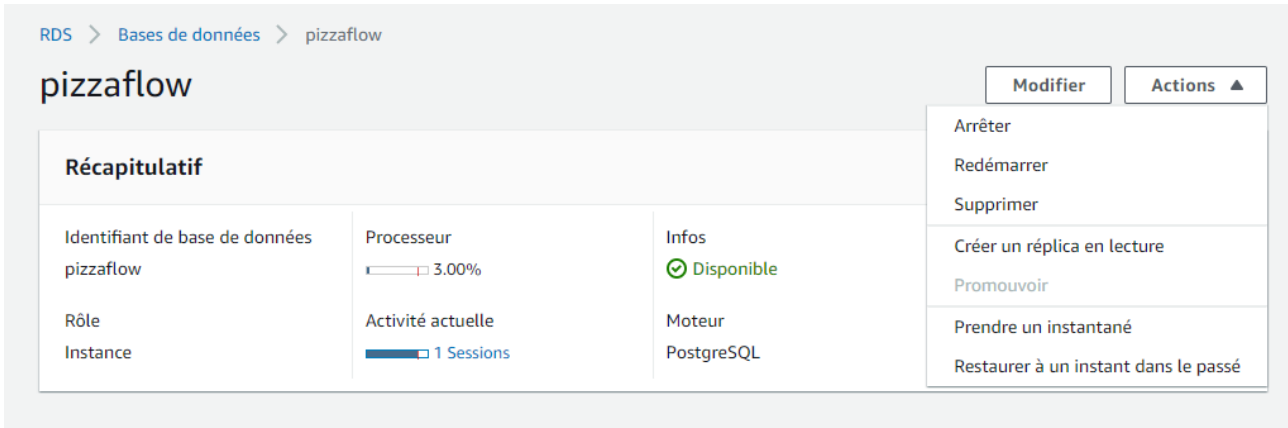
### 5.1.2 - Démarrage

Dans le menu "**Action**" en haut à droite sélectionner "**Redémarrer**".



### 5.1.3 - Arrêt

Dans le menu "**Action**" en haut à droite sélectionner "**Arrêter**".



The screenshot shows the AWS RDS console for a database instance named 'pizzaflow'. The breadcrumb navigation is 'RDS > Bases de données > pizzaflow'. The instance details are as follows:

Récapitulatif		
Identifiant de base de données pizzaflow	Processeur 3.00%	Infos ✓ Disponible
Rôle Instance	Activité actuelle 1 Sessions	Moteur PostgreSQL

The 'Actions' dropdown menu is open, showing the following options: Arrêter, Redémarrer, Supprimer, Créer un réplica en lecture, Promouvoir, Prendre un instantané, and Restaurer à un instant dans le passé.

## 5.2 - Base de données Stock

Effectuer les mêmes opérations que pour la base de données **User** ci-dessus.

## 5.3 - Base de données Gestion

Effectuer les mêmes opérations que pour la base de données **User** ci-dessus.

## 5.4 - Config-server

### 5.4.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **config-server** et se placer dans le répertoire de l'application.

```
$cd /srv/config-server
```

### 5.4.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/config-server$ sudo ./start.sh
```

### 5.4.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/config-server$ sudo ./shutdown.sh
```

## 5.5 - user-api

### 5.5.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **user-api** et se placer dans le répertoire de l'application.

```
$ cd /srv/user-api
```

### 5.5.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/user-api$ sudo ./start.sh
```

### 5.5.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/user-api$sudo ./shutdown.sh
```

## 5.6 - stock-api

### 5.6.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **stock-api** et se placer dans le répertoire de l'application.

```
$cd /srv/stock-api
```

### 5.6.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/stock-api$sudo ./start.sh
```

### 5.6.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/stock-api$sudo ./shutdown.sh
```

## 5.7 - web-api

### 5.7.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **web-api** et se placer dans le répertoire de l'application.

```
$cd /srv/web-api
```

### 5.7.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/web-api$sudo ./start.sh
```

### 5.7.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/web-api$sudo ./shutdown.sh
```

## 5.8 - production-api

### 5.8.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **production-api** et se placer dans le répertoire de l'application.

```
$cd /srv/production-api
```

### 5.8.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/production-api$sudo ./start.sh
```

### 5.8.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/production-api$sudo ./shutdown.sh
```

## 5.9 - gestion-api

### 5.9.1 - Préalable

Se connecter avec **PuTTY** sur la **VM** de **gestion-api** et se placer dans le répertoire de l'application.

```
$cd /srv/gestion-api
```

### 5.9.2 - Démarrage

Exécuter le script de démarrage.

```
/srv/gestion-api$sudo ./start.sh
```

### 5.9.3 - Arrêt

Exécuter le script d'arrêt.

```
/srv/gestion-api$sudo ./shutdown.sh
```



## 6 - PROCÉDURE DE MISE À JOUR

Les mises à jour doivent se faire uniquement en période creuse et de préférence la nuit pour ne pas nuire aux utilisateurs et pour avoir assez de temps pour effectuer les opérations

### 6.1 - Base de données User

#### 6.1.1 - Préalable

Récupérer le script **SQL** de mise à jour de la base de données **User** sur son ordinateur pour pouvoir l'exécuter dans **pgAdmin**.

Arrêter le microservice **user-api** suivant la procédure indiquée plus haut pour qu'ils n'accèdent pas à la base de données.

Faire une sauvegarde de la base de données comme indiqué dans les procédures de sauvegarde et de restauration.

#### 6.1.2 - Mise à jour

Depuis **pgAdmin** se connecter à la base de données **User**. Dans la fenêtre éditeur de requêtes ouvrir le script **SQL** récupéré et l'exécuter comme pour l'ajout des données dans la base lors du déploiement.

#### 6.1.3 - Finalisation

Relancer le microservice **user-api**.

### 6.2 - Base de données Stock

#### 6.2.1 - Préalable

Récupérer le script **SQL** de mise à jour de la base de données **Stock** sur son ordinateur pour pouvoir l'exécuter dans **pgAdmin**.

Arrêter le microservice **stock-api** suivant la procédure indiquée plus haut pour qu'ils n'accèdent pas à la base de données.

Faire une sauvegarde de la base de données comme indiqué dans les procédures de sauvegarde et de restauration.

### 6.2.2 - Mise à jour

Depuis **pgAdmin** se connecter à la base de données **Stock**. Dans la fenêtre éditeur de requêtes ouvrir le script **SQL** récupéré et l'exécuter comme pour l'ajout des données dans la base lors du déploiement.

### 6.2.3 - Finalisation

Relancer le microservice **stock-api**.

## 6.3 - Base de données Gestion

### 6.3.1 - Préalable

Récupérer le script **SQL** de mise à jour de la base de données **Gestion** sur son ordinateur pour pouvoir l'exécuter dans **pgAdmin**.

Arrêter le microservice **gestion-api** suivant la procédure indiquée plus haut pour qu'ils n'accèdent pas à la base de données.

Faire une sauvegarde de la base de données comme indiqué dans les procédures de sauvegarde et de restauration.

### 6.3.2 - Mise à jour

Depuis **pgAdmin** se connecter à la base de données **Gestion**. Dans la fenêtre éditeur de requêtes ouvrir le script **SQL** récupéré et l'exécuter comme pour l'ajout des données dans la base lors du déploiement.

### 6.3.3 - Finalisation

Relancer le microservice **gestion-api**.

## 6.4 - Microservice user-api

### 6.4.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **user-api** avec les commandes suivantes :

```
$cd /srv/user-api  
/srv/user-api$sudo ./shutdown.sh
```

Effectuer une sauvegarde de l'instance sur **AWS** en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.4.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **user-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/user-api$ wget -O user-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect?  
\  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= user-api \  
&v=2.1 \  
&p=zip  
/srv/user-api$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.4.3 - Finalisation

Relancer le microservice **user-api**.

## 6.5 - Microservice stock-api

### 6.5.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **stock-api** avec les commandes suivantes :

```
$cd /srv/stock-api  
/srv/stock-api$sudo ./shutdown.sh
```

Effectuer une sauvegarde de l'instance sur **AWS** en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.5.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **stock-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/stock-api$          wget          -O          stock-api.zip          \  
http://itcd.com/nexus/service/local/artifact/maven/redirect? \  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= stock-api \  
&v=2.1 \  
&p=zip  
/srv/stock-api$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.5.3 - Finalisation

Relancer le microservice **stock-api**.

## 6.6 - Microservice web-api

### 6.6.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **web-api** avec les commandes suivantes :

```
$cd /srv/web-api  
/srv/web-api$sudo ./shutdown.sh
```

Effectuer une sauvegarde de l'instance sur **AWS** en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.6.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **web-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/web-api$ wget -O web-api.zip \ http://itcd.com/nexus/service/local/artifact/maven/redirect?  
\  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= web-api \  
&v=2.1 \  
&p=zip  
/srv/web-api$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.6.3 - Finalisation

Relancer le microservice **web-api**.

## 6.7 - Microservice production-api

### 6.7.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **production-api** avec les commandes suivantes :

```
$cd /srv/production-api  
/srv/production-api$sudo ./shutdown.sh
```

Effectuer une sauvegarde du microservice en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.7.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **production-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/production-api$ wget -O production-api.zip \  
http://itcd.com/nexus/service/local/artifact/maven/redirect? \  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= production-api \  
&v=2.1 \
```

```
&p=zip  
/srv/production-api$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.7.3 - Finalisation

Relancer le microservice **production-api**.

## 6.8 - Microservice gestion-api

### 6.8.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **gestion-api** avec les commandes suivantes :

```
$cd /srv/gestion-api  
/srv/gestion-api$sudo ./shutdown.sh
```

Effectuer une sauvegarde du microservice en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.8.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **gestion-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/gestion-api$ wget -O gestion-api.zip \  
http://itcd.com/nexus/service/local/artifact/maven/redirect? \  
r=pizzaflow \  
&g=com.itcd.delivery \  
&a= gestion-api \
```

```
&v=2.1 \  
&p=zip  
/srv/gestion-api$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.8.3 - Finalisation

Relancer le microservice **gestion-api**.

## 6.9 - Serveur de configuration

### 6.9.1 - Préalable

Aller dans le répertoire de l'application et arrêter le microservice **config-server** avec les commandes suivantes :

```
$cd /srv/config-server  
/srv/config-server$sudo ./shutdown.sh
```

Effectuer une sauvegarde du microservice en suivant la procédure indiquée dans les sauvegardes des microservices.

### 6.9.2 - Mise à jour

Récupérer le **ZIP** de mise à jour en spécifiant le bon numéro de version (v=?) de **web-api** qui est dans un repository **Nexus** et lancer la mise à jour avec les commandes :

```
/srv/config-server$          wget          -O          config-server.zip          \  
http://itcd.com/nexus/service/local/artifact/maven/redirect? \  
r=pizzaflow \  

```



```
&g=com.itcd.delivery \  
&a= config-server \  
&v=2.1 \  
&p=zip  
/srt/config-server$sudo ./update.sh
```

Le script **update.sh** arrête le microservice si c'est nécessaire, effectue une sauvegarde des fichiers de l'application et décompresse l'archive **ZIP**.

### 6.9.3 - Finalisation

Relancer le microservice **config-server**.

## 7 - SUPERVISION/MONITORING

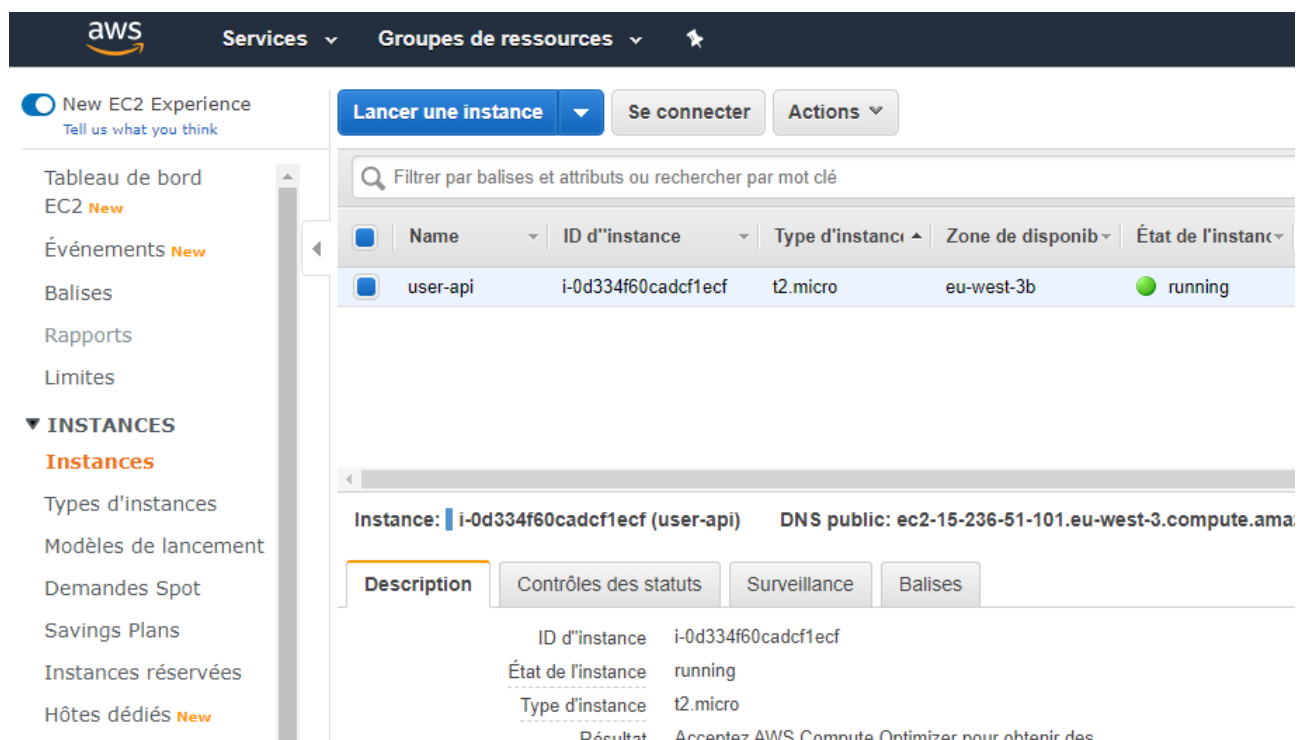
La supervision et le monitoring se font à plusieurs niveaux :

- Au niveau matériel avec les indicateurs des tableaux de bord des instances sur **AWS**.
- Supervision des métriques exposées par **actuator** sur **AWS CloudWatch**.
- Analyser les logs en temps réel avec **Kibana**.
- Remontée des sondes sur un compte privé **Twitter**.

### 7.1 - Supervision hardware des instances des microservices

Lancer un navigateur à l'adresse [www.pizzaflow.com](http://www.pizzaflow.com) pour vérifier que la page d'accueil s'affiche correctement.

Se rendre sur l'application de gestion des instances **EC2** sur **AWS** et sélectionner l'écran **"INSTANCES/Instances"** pour voir l'état des instances.



**aws** Services ▾ Groupes de ressources ▾

New EC2 Experience  
Tell us what you think

Tableau de bord  
EC2 **New**  
Événements **New**  
Balises  
Rapports  
Limites  
▼ **INSTANCES**  
**Instances**  
Types d'instances  
Modèles de lancement  
Demandes Spot  
Savings Plans  
Instances réservées  
Hôtes dédiés **New**

Lancer une instance ▾ Se connecter Actions ▾

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name	ID d'instance	Type d'instance	Zone de disponibilité	État de l'instance
<input type="checkbox"/>	user-api	i-0d334f60cadcf1ecf	t2.micro	eu-west-3b	● running

Instance: **i-0d334f60cadcf1ecf (user-api)** DNS public: ec2-15-236-51-101.eu-west-3.compute.amazonaws.com

Description Contrôles des statuts Surveillance Balises

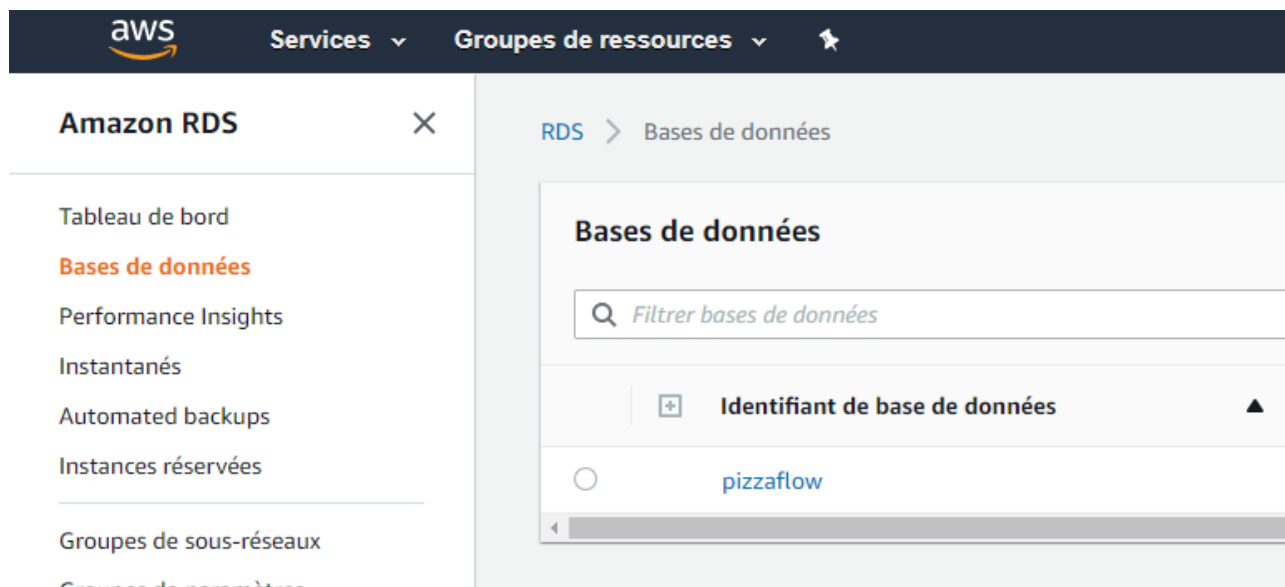
ID d'instance i-0d334f60cadcf1ecf  
État de l'instance running  
Type d'instance t2.micro  
Résultat Acceptez AWS Compute Optimizer pour obtenir des

L'état des instances doit être indiqué en vert comme dans l'exemple ci-dessus avec **user-api**. Sélectionner l'instance voulue puis l'onglet "Surveillance" pour voir le tableau de monitoring de l'instance.



## 7.2 - Supervision hardware des instances des bases de données

Se rendre sur l'application de gestion **Amazon RDS** et sélectionner l'écran "**Base de données**" pour voir l'état des bases.



Sélectionner une base de données puis l'onglet "**Surveillance**" pour voir le tableau de monitoring de la base choisie.

RDS &gt; Bases de données &gt; pizzaflow

## pizzaflow

### Récapitulatif

Identifiant de base de données  
pizzaflow

Rôle  
Instance

Processeur  
4.17%

Activité actuelle  
1 Sessions

Infos  
Disponible

Moteur  
PostgreSQL

Connectivité et sécurité

Surveillance

Journaux et événements

Configuration

Maintenance et sauvegardes

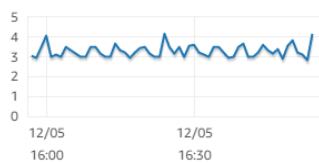
Balises

### CloudWatch (20)

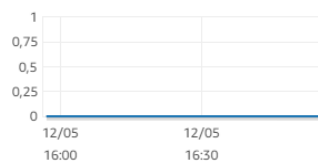
Légende : pizzaflow

Q

#### Utilisation de l'UC (Pourcentage)



#### Connections DB (Nombre)

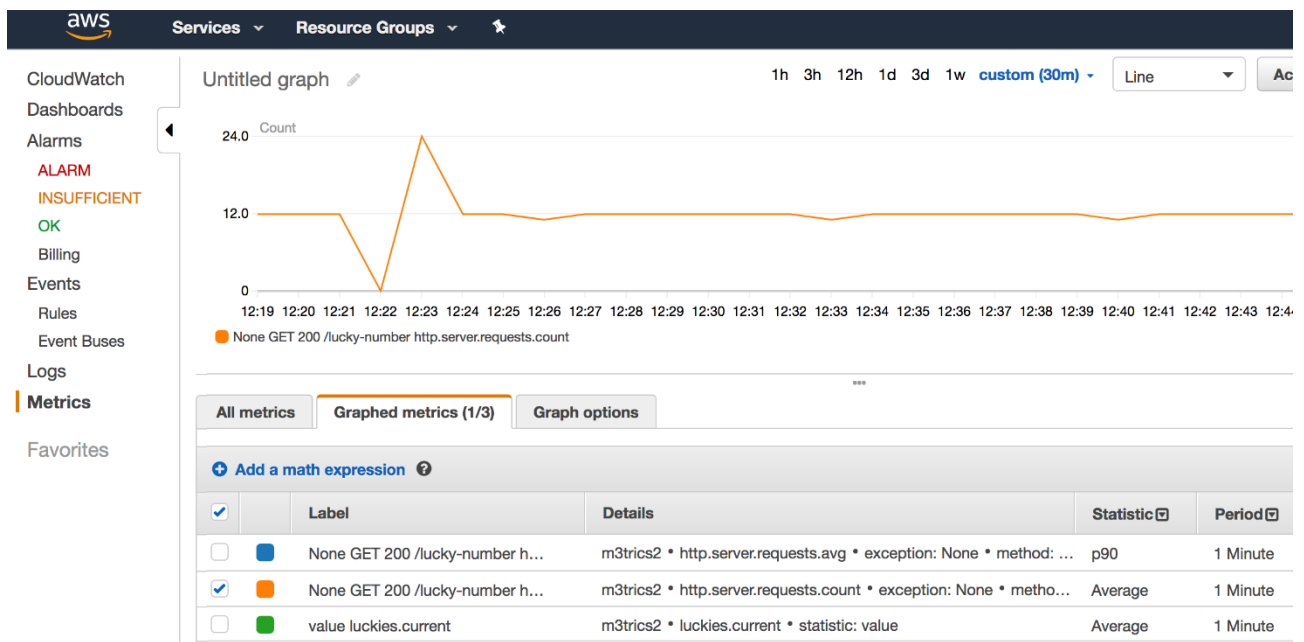
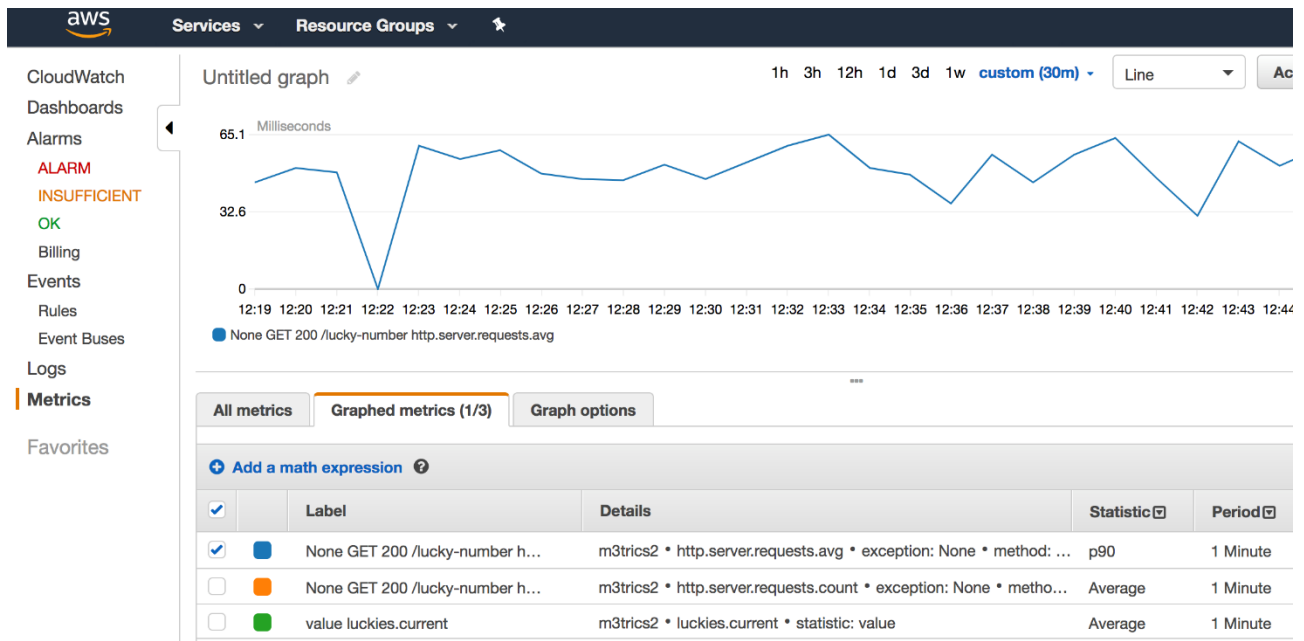


#### Espace de stockage disponible (Mo)



## 7.3 - CloudWatch Metrics

L'application utilise **Spring Boot Actuator** et le starter **Spring Cloud AWS** pour récupérer les métriques sur un tableau de monitoring d'**AWS**. Se connecter sur l'interface d'**AWS** des instances des microservices et sélectionner l'écran "**Metrics**".



## 7.4 - Analyse des logs

La seule façon de se rendre compte qu'une application fonctionne est d'analyser les logs qu'elle produit en temps réel. On peut suivre les temps de réponses, les codes de retours et le trafic sur un dashboard.

- **Elasticsearch** est la base de données **NoSQL** et moteur de recherche.
- **Logstash** est l'interface entre les données de logs générées et **Elasticsearch**.
- **Kibana** est le tableau de bord qui filtre les données venant d'**Elasticsearch** en temps réel.

Ce service est hébergé **logit.io** pour être découplé et indépendant de notre système applicatif.

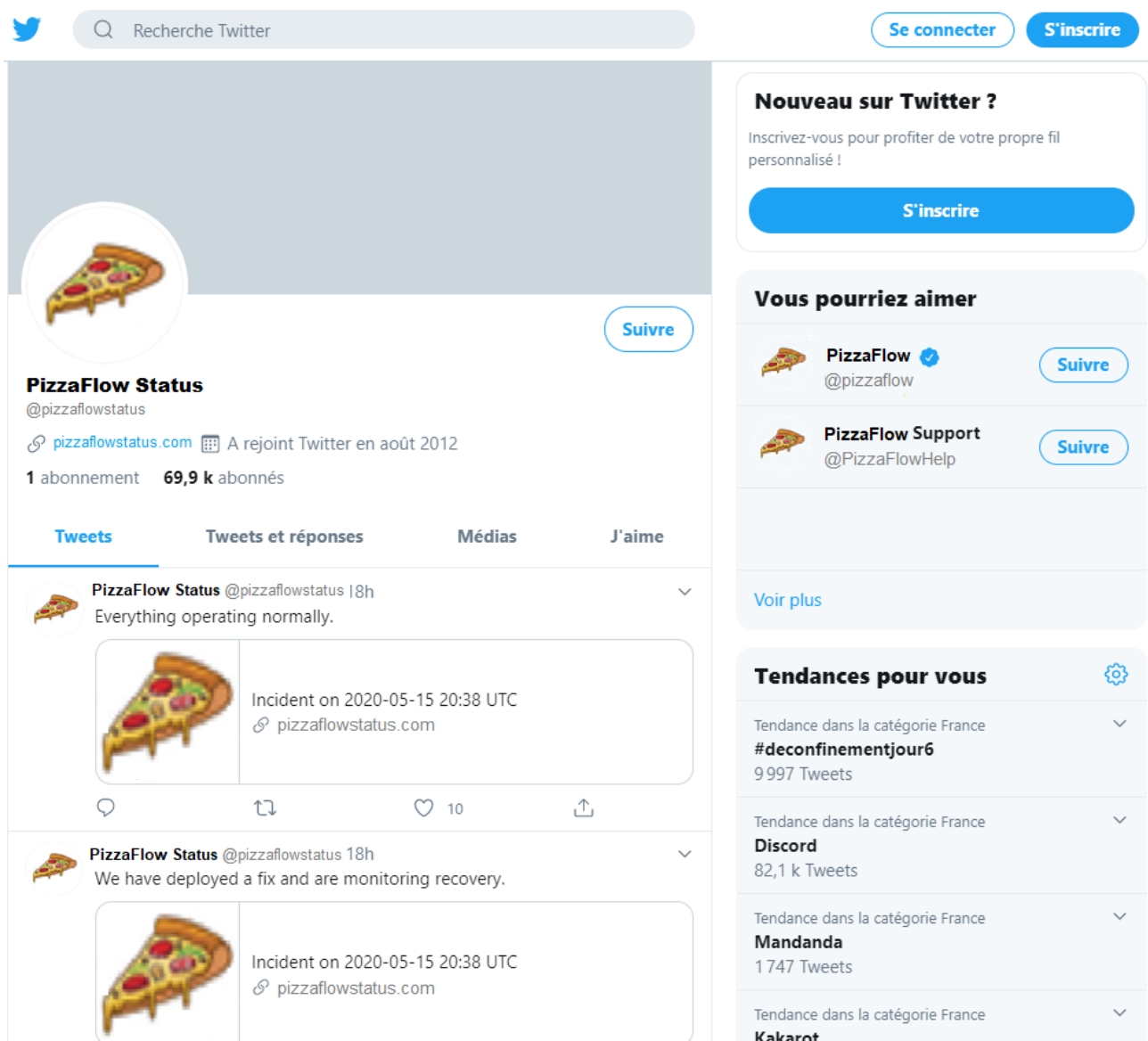


## 7.5 - Les sondes

Pour vérifier que l'application est toujours fonctionnelle et que la base de données n'a pas de problème, des batch effectuent des requêtes sur les microservices et comparent avec le résultat attendu.

La différence avec le résultat attendu est interprétée est une alarme est envoyées via **Twitter**. Un message de retour au service est envoyé sur **Twitter**.

Avec ce système de sonde, les responsables sont avertis dès qu'une défaillance apparaît.



The screenshot shows the Twitter profile of **PizzaFlow Status** (@pizzaflowstatus). The profile includes a pizza icon, the name **PizzaFlow Status**, the handle @pizzaflowstatus, a link to pizzaflowstatus.com, and a bio stating "A rejoint Twitter en août 2012". It shows 1 abonnement and 69,9 k abonnés. The main content area displays two tweets: the first says "Everything operating normally." and the second says "We have deployed a fix and are monitoring recovery." Both tweets include a pizza icon and a link to pizzaflowstatus.com. The right sidebar features a "Nouveau sur Twitter ?" section with a "S'inscrire" button, a "Vous pourriez aimer" section with links to PizzaFlow and PizzaFlow Support, and a "Tendances pour vous" section listing trending topics like #deconfinementjour6, Discord, Mandanda, and Kakarot.

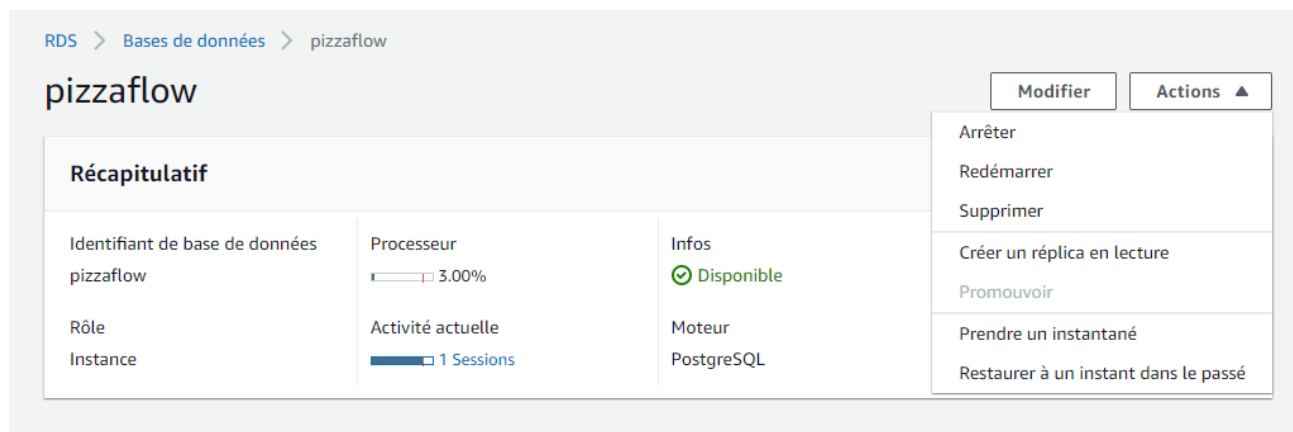


## 8 - PROCÉDURE DE SAUVEGARDE ET RESTAURATION

### 8.1 - Base de données

#### 8.1.1 - Sauvegarde d'une base de données

Par sécurité il faut toujours faire une sauvegarde de l'état de la base de données pour pouvoir revenir en arrière en cas de problème de mise à jour. Depuis le menu "Action" de l'interface **Amazon RDS** de la base de données choisie sélectionner "Prendre un instantané".




The screenshot shows the Amazon RDS console interface for a database instance named 'pizzaflow'. The breadcrumb navigation at the top reads 'RDS > Bases de données > pizzaflow'. The instance name 'pizzaflow' is prominently displayed. Below it, a 'Récapitulatif' (Summary) section provides key details:

- Identifiant de base de données:** pizzaflow
- Rôle:** Instance
- Processeur:** 3.00% (indicated by a progress bar)
- Activité actuelle:** 1 Sessions (indicated by a progress bar)
- Infos:** Disponible (with a green checkmark icon)
- Moteur:** PostgreSQL

To the right of the summary, there are two buttons: 'Modifier' and 'Actions'. The 'Actions' dropdown menu is open, showing several options: 'Arrêter', 'Redémarrer', 'Supprimer', 'Créer un réplica en lecture', 'Promouvoir', 'Prendre un instantané' (which is highlighted), and 'Restaurer à un instant dans le passé'.

Dans la fenêtre qui s'ouvre on spécifie le nom de l'instantané en respectant l'expression :

**[nom-base]YYYYMMDD-HHMM**


Services ▾ Groupes de ressources ▾

RDS > Bases de données > pizzaflow > Prendre un instantané

## Capture d'un instantané DB

### Paramètres

Pour prendre un instantané de cette instance DB, vous devez fournir un nom pour l'instantané.

**Instance DB**  
La clé unique qui identifie une instance DB. Ce paramètre n'est pas sensible à la casse.  
pizzaflow

**Nom de l'instantané**  
Identificateur pour l'instantané DB.

Annuler **Prendre un instantané**

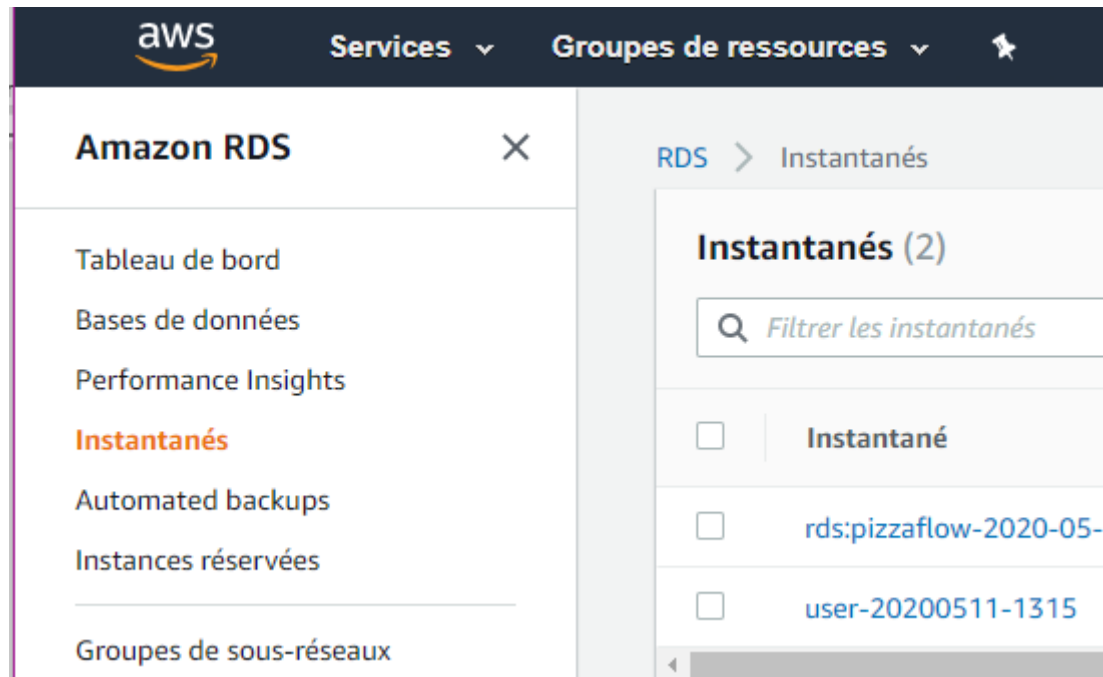
Il apparaît ensuite dans la liste des instantanés de la base.

Instantanés (2)		
<input type="text" value="Filtrer les instantanés"/>		
<input type="checkbox"/>	Instantané ▾	Instance DB ou cluster ▾
<input type="checkbox"/>	rds:pizzaflow-2020-05-12-08-27	pizzaflow
<input type="checkbox"/>	user-20200511-1315	pizzaflow

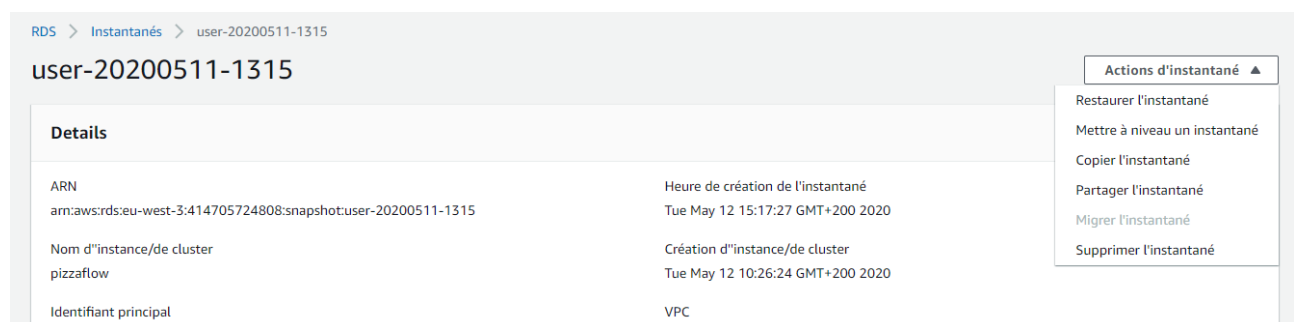
Il suffira de le sélectionner l'instantané pour restaurer la base.

## 8.1.2 - Restauration d'une base de données

Dans l'arborescence à gauche de l'interface **Amazon RDS** de la base de données sélectionner **"Instantanés"**.



Sélectionner ensuite l'instantané de la base voulu et choisir **"Restaurer l'instantané"** dans le menu **"Action d'instantané"**.



La base de données est restaurée suivant l'instantané choisi.

## 8.2 - Microservices

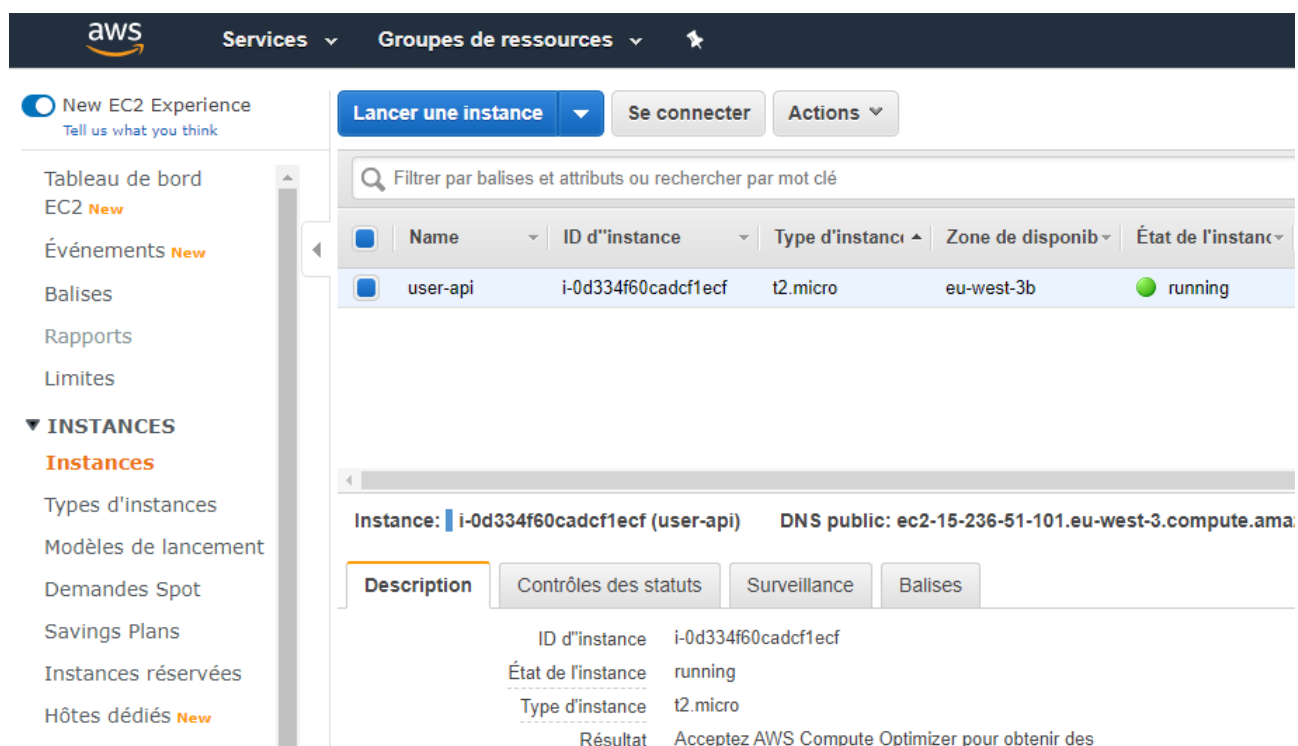
### 8.2.1 - Sauvegarde

La sauvegarde d'un microservice se fait en deux temps :

- Prise d'un instantané de l'instance EC2.
- Backup des fichiers du microservice.

#### 8.2.1.1 - Sauvegarde sur AWS.

Une sauvegarde des instances sous forme d'image est effectuée tous les jours avec une profondeur de 10 jours. Pour effectuer une sauvegarde manuelle, se rendre sur l'application de gestion des instances **EC2** sur **AWS** et sélectionner l'écran "**INSTANCES/Instances**" puis l'instance à sauvegarder dans l'écran de gauche.



The screenshot displays the AWS Management Console interface for EC2 instances. The left-hand navigation pane shows the 'INSTANCES' section expanded, with 'Instances' highlighted. The main content area shows a table of instances. The table has columns for Name, ID d'instance, Type d'instance, Zone de disponibilité, and État de l'instance. One instance, 'user-api', is listed with ID 'i-0d334f60cadcf1ecf', type 't2.micro', and state 'running'. Below the table, the details for the selected instance are shown, including its ID, state, type, and a recommendation to use AWS Compute Optimizer.

Name	ID d'instance	Type d'instance	Zone de disponibilité	État de l'instance
user-api	i-0d334f60cadcf1ecf	t2.micro	eu-west-3b	running

Instance: **i-0d334f60cadcf1ecf (user-api)** DNS public: **ec2-15-236-51-101.eu-west-3.compute.amazonaws.com**

Description	Contrôles des statuts	Surveillance	Balises
ID d'instance	i-0d334f60cadcf1ecf		
État de l'instance	running		
Type d'instance	t2.micro		
Résultat	Acceptez AWS Compute Optimizer pour obtenir des		

Sélectionner dans le menu "**Actions**" le sous-menu "**Image/Créer l'image**".

Lancer une instance ▼ Se connecter Actions ^

Filter par balises et attributs ou rechercher par

Name	ID d'instance	État de l'instance	Contrôles des s	Statut des
user-api	i-0d334f60cadcf1ecf	running	2/2 contrôle...	Aucun(e)

Se connecter  
Obtenir le mot de passe de Windows  
Create Template From Instance  
En lancer plus comme ceci

État de l'instance  
Paramètres de l'instance  
Image  
Mise en réseau  
Supervision de CloudWatch

Créer l'image  
Instance de groupe (AMI de stockage d'instance)

Indiquer le nom de l'image suivant la notation suivante et créer l'image :

**[nom-microservice]-YYYYMMDD-HHMM**

### Créer l'image

ID d'instance ⓘ i-0d334f60cadcf1ecf

Nom de l'image ⓘ

Description de l'image ⓘ

Pas de redémarrage ⓘ ☒

Volumes d'instance

Type de volume ⓘ	Dispositif ⓘ	Instantané ⓘ	Taille (Gio) ⓘ	Type de volume ⓘ	IOPS ⓘ	Débit (Mbit/s) ⓘ	Supprimer à la résiliation ⓘ	Chiffré ⓘ
Racine	/dev/xvda	snap-0850d0da10e0060fb	<input type="text" value="8"/>	Volume à usage général SSD (gp2) ▼	100 / 3000	N/A	<input checked="" type="checkbox"/>	Non chiffré

[Ajouter un nouveau volume](#)

Taille totale des volumes EBS : 8 Gio  
Lorsque vous créez une image EBS, un instantané EBS sera également créé pour chacun des volumes ci-dessus.

[Annuler](#) [Créer l'image](#)

Vérifier que l'image a été créée et apparait dans la liste en sélectionnant dans l'arborescence de gauche le menu **"IMAGES/AMI"** avec son nom, son statut et sa date de création

New EC2 Experience  
Tell us what you think

Événements **New**

Balises

Rapports

Limites

▼ INSTANCES

Instances

Types d'instances

Modèles de lancement

Demandes Spot

Savings Plans

Instances réservées

Hôtes dédiés **New**

Réservations de capacité

▼ IMAGES

**AMI**

Lancer Actions

M'appartenant

Filtrer par balises et attributs ou rechercher par mot clé

Name	Nom d'AMI	ID d'AMI	Source	Propriétaire	Visibilité	Statut	Date de création
user-api-2020...	ami-018f01b2d282db499	414705724808/...	414705724808	Privé	available	16 mai 2020 15:28:30	

### 8.2.1.2 - Backup.

Se connecter à l'instance du microservice choisi avec **PuTTY**. Aller dans le répertoire de l'application avec la commande suivante en modifiant la valeur entre crochets par le nom du microservice et lancer le script **backup.sh**.

```
$cd /srv/[nom-microservice]
/srv/[nom-microservice]$sudo ./backup.sh
```

Le script effectue une sauvegarde dans le répertoire dont le nom est au format :

**/srv/[nom-microservice]/backup/[nom-microservice]-YYYYMMDD-HHMM**

Et copie un instantané du microservice. Relancer le microservice avec la commande :

```
$cd /srv/[nom-microservice]
/srv/[nom-microservice]$sudo ./start.sh
```

### 8.2.2 - Restauration ancienne configuration

Deux types de restaurations sont disponibles :

- Restauration des binaires de l'application à une version précédente.
- Restauration du système sous **AWS** à partir d'une image.

Une sauvegarde automatique journalière des instances est effectuée sur **AWS** avec un archivage des 10 dernières images. On peut ajouter des sauvegardes en créant des images manuellement.

### 8.2.2.1 - Restauration des binaires

Se connecter à l'instance du microservice choisi avec **PuTTY**. Aller dans le répertoire de l'application et arrêter le microservice avec les commandes suivantes en modifiant la valeur entre crochets par le nom du microservice.

```
$cd /srv/[nom_microservice]  
/srv/[nom_microservice]$sudo ./shutdown.sh
```

Lancer le script de restauration en spécifiant la date **YYYYMMDD** et l'heure **HHMM** du système à restaurer avec la commande suivante :

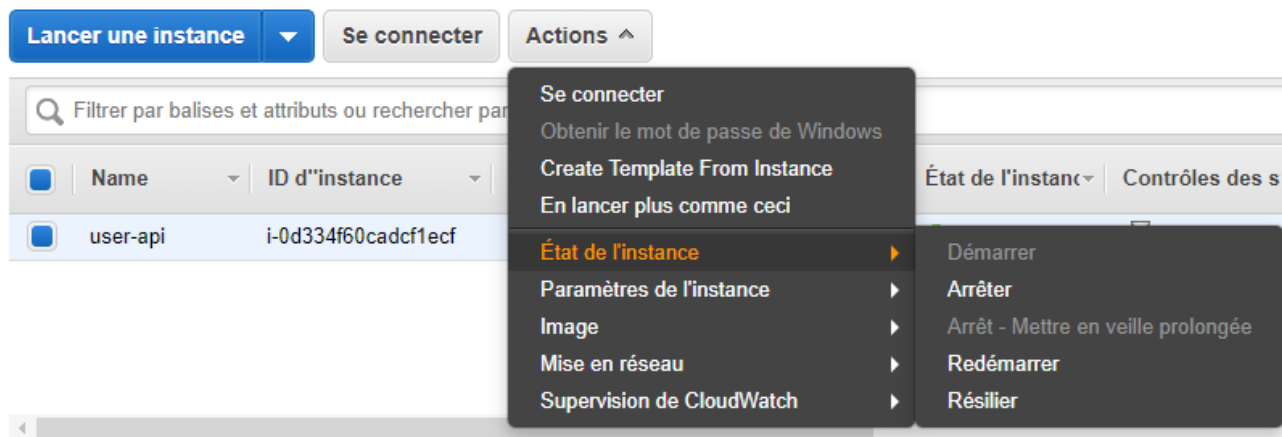
```
/srv/[nom_microservice]$sudo ./restore.sh [YYYYMMDD]-[HHMM]
```

La configuration dans le backup **nom-microservice-YYYYMMDD-HHMM** est restaurée. Relancer le microservice avec la commande :

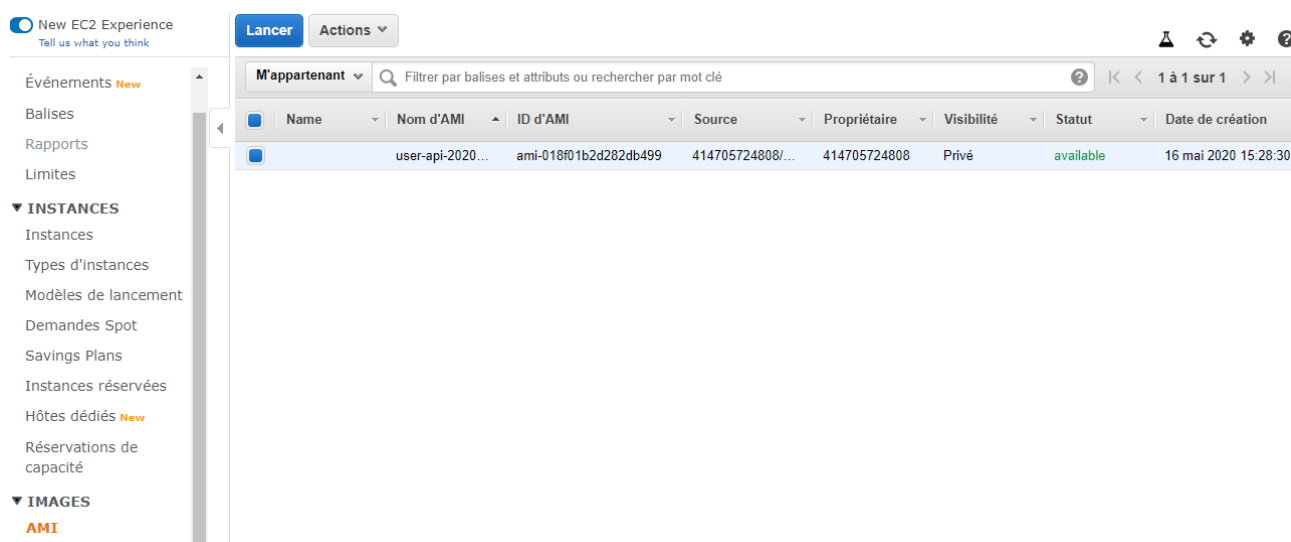
```
/srv/[nom_microservice]$sudo ./start.sh
```

### 8.2.2.2 - Restauration d'une image d'instance sur AWS

Pour effectuer une restauration manuelle d'une instance à l'aide de son image, se rendre sur l'application de gestion des instances **EC2** sur **AWS** pour arrêter l'instance en cours. Sélectionner l'instance puis dans le menu "**Actions**" le sous-menu "**État de l'instance/Arrêter**".

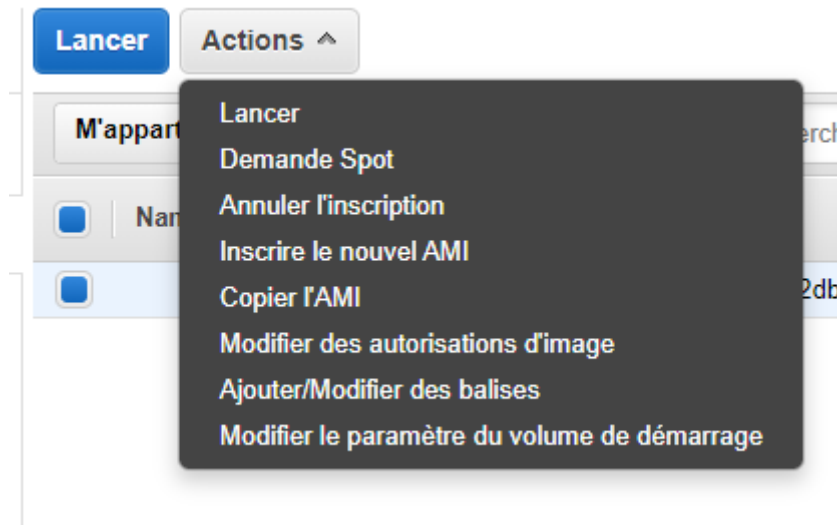


Sélectionner l'écran "**IMAGES/AMI**" puis l'image de l'instance à restaurer dans l'écran de gauche.



Dans le menu "**Action**" sélectionner "**Lancer**".





Dans les écrans suivants vérifier et valider.


**aws** Services ▾ Groupes de ressources ▾

1. Choisir l'AMI 2. Choisir un type d'instance 3. Configurer l'instance 4. Ajouter le stockage 5. Ajouter des balises 6. Configurer le groupe de sécurité 7. Vérification

### Étape 7 : Examiner le lancement de l'instance

Veillez vérifier les détails de votre lancement d'instance. Vous pouvez revenir en arrière pour modifier les changements pour chaque section. Cliquez sur **Lancer** pour affecter une paire de clés à votre instance et terminer la procédure de lancement.

▼ Détails de l'AMI [Modifier l'AMI](#)

 **user-api-20200516-1525 - ami-018f01b2d282db499**  
Sauvegarde avant MAJ  
Type de périphérique racine: ebs Type de virtualisation: hvm

▼ Type d'instance [Modifier le type d'instance](#)

Type d'instance	ECU	vCPU	Mémoire (Go)	Stockage d'instance (Go)	Disponible en version optimisée pour EBS	Performances réseau
t2.micro	Variable	1	1	EBS uniquement	-	Low to Moderate

▼ Groupes de sécurité [Modifier les groupes de sécurité](#)

**Nom du groupe de sécurité** launch-wizard-5  
**Description** launch-wizard-5 created 2020-05-16T15:54:53.753+02:00

Type ⓘ	Protocole ⓘ	Plage de ports ⓘ	Source ⓘ	Description ⓘ
Ce groupe de sécurité n'a pas de règle				

► Détails de l'instance [Modifier les détails de l'instance](#)

► Stockage [Modifier le stockage](#)

► Balises [Modifier les balises](#)

[Annuler](#) [Précédent](#) [Lancer](#)

Vérifier les clés et valider pour afficher l'écran des instances.

Lancer une instance

▼

Se connecter

Actions ▼

🔍

Filtrer par balises et attributs ou rechercher par mot clé

<input type="checkbox"/>	Name ▼	ID d'instance ▼	Type d'instance ▲	Zone de disponib ▼	État de l'instanc ▼
<input type="checkbox"/>		i-0bf44e66f9f07080a	t2.micro	eu-west-3b	<div>●</div> running
<input checked="" type="checkbox"/>	user-api	i-0d334f60cadcf1ecf	t2.micro	eu-west-3b	<div>●</div> stopped

Après vérification du fonctionnement de l'application, on peut supprimer l'ancienne instance qui arrêtée en choisissant **"État de l'instance/Résilier"** dans le menu **"Actions"**.

## 9 - GLOSSAIRE

<b>AMI</b>	( <b>Amazon Machine Image</b> ) logiciel d'exploitation Amazone de type linux.
<b>AWS</b>	( <b>Amazon Web Services</b> ) service internet d'Amazon.
<b>CNIL</b>	( <b>Commission nationale de l'informatique et des libertés</b> ).
<b>CSS</b>	( <b>Cascading Style Sheets</b> ) fichier de style pour la présentation des pages <b>HTML</b> .
<b>DTO</b>	( <b>Data Transfer Object</b> ) type d'objet permettant de transférer des données.
<b>EC2</b>	( <b>Elastic Cloud Compute</b> ) serveur de base permettant d'intégrer de nombreux systèmes.
<b>IAM</b>	( <b>Identity and Access Management</b> ) service d'Amazon pour gérer l'authentification des utilisateurs.
<b>Jasypt</b>	( <b>Java Simplified encryption</b> ) librairie java qui permet d'effectuer un cryptage basic dans des fichiers de configuration.
<b>JDK</b>	( <b>Java Developer Kit</b> ) outils de développement du langage <b>Java</b> .
<b>JRE</b>	( <b>Java Runtime Environment</b> ) outils pour exécuter un exécutable <b>Java</b> .
<b>JS</b>	<b>Javascript</b> est un langage de script pour les pages web.
<b>JSON</b>	( <b>JavaScript Object Notation</b> ) format léger d'échange de données facilement compréhensible par l'homme et manipulable par l'ordinateur.
<b>load-balancing</b>	Action de répartir la charge entre plusieurs instances d'une même application.
<b>NoSQL</b>	( <b>Not Only SQL</b> ) Type de base de données qui n'utilise pas l'architecture classique des bases de données relationnelles <b>SQL</b> .
<b>pgAdmin</b>	( <b>PostgreSQL Admin</b> ) logiciel d'administration de bases de données.
<b>PPK</b>	( <b>PuTTY Private Key</b> ) Fichier de clé privée pour <b>PuTTY</b> .
<b>PuTTY</b>	Logiciel pour se connecter à distance sur une machine.
<b>RDS</b>	( <b>Relational Database Service</b> ) Serveur avec un <b>SGBD-R</b> intégré.
<b>Repository</b>	Répertoire dans le cloud.
<b>S3</b>	( <b>Simple storage Service</b> ) serveur de fichiers static pour les sites web.
<b>SGBD-R</b>	( <b>Système de Gestion de Bases de Données Relationnelles</b> ).
<b>SLF4J</b>	( <b>Simple Loggin Facade for Java</b> ) couche abstraite pour l'utilisation de différents loggers.
<b>VM</b>	( <b>Virtual Machine</b> ) machine virtuelle qui contient un système d'exploitation pour faire fonctionner une application.