

OC Pizza

PizzaFlow

Dossier d'exploitation

Version 1.0

Auteur

Paul-Emmanuel DOS SANTOS FACAO
Analyste programmeur

TABLE DES MATIÈRES

| | |
|---|----------|
| 1 - Versions..... | 6 |
| 2 - Introduction | 7 |
| 2.1 - Objet du document | 7 |
| 2.2 - Références..... | 7 |
| 3 - Pré-requis | 8 |
| 3.1 - Création d'une paire de clé | 8 |
| 3.2 - Système..... | 10 |
| 3.2.1 - <i>Serveur de Base de données User</i> | 10 |
| 3.2.1.1 - Description | 10 |
| 3.2.1.2 - Caractéristiques techniques | 10 |
| 3.2.2 - <i>Serveur de Base de données Stock</i> | 11 |
| 3.2.2.1 - Description | 11 |
| 3.2.2.2 - Caractéristiques techniques | 12 |
| 3.2.3 - <i>Serveur de Base de données Gestion</i> | 13 |
| 3.2.3.1 - Description | 13 |
| 3.2.3.2 - Caractéristiques techniques | 13 |
| 3.2.4 - <i>Serveur de configuration</i> | 14 |
| 3.2.4.1 - Description | 14 |
| 3.2.4.2 - Caractéristiques techniques | 15 |
| 3.2.5 - <i>user-api</i> | 16 |
| 3.2.5.1 - Description | 16 |
| 3.2.5.2 - Caractéristiques techniques | 16 |
| 3.2.6 - <i>stock-api</i> | 17 |
| 3.2.6.1 - Description | 17 |
| 3.2.6.2 - Caractéristiques techniques | 18 |
| 3.2.7 - <i>web-api</i> | 19 |
| 3.2.7.1 - Description | 19 |
| 3.2.7.2 - Caractéristiques techniques | 19 |
| 3.2.8 - <i>production-api</i> | 20 |
| 3.2.8.1 - Description | 20 |
| 3.2.8.2 - Caractéristiques techniques | 21 |
| 3.2.9 - <i>gestion-api</i> | 22 |
| 3.2.9.1 - Description | 22 |
| 3.2.9.2 - Caractéristiques techniques | 22 |
| 3.2.10 - <i>Serveur de Fichiers</i> | 22 |
| 3.2.10.1 - Description..... | 22 |
| 3.3 - Bases de données..... | 23 |
| 3.4 - Les contrôleurs..... | 23 |
| 3.5 - Web-services | 24 |
| 3.6 - Serveur de configuration..... | 24 |
| 3.7 - Les fichiers statiques | 25 |

| | |
|---|-----------|
| 4 - Procédure de déploiement | 26 |
| 4.1 - Configuration de pgAdmin..... | 26 |
| 4.2 - Création du schéma de la base User..... | 28 |
| 4.3 - Insérer les données initiales de la base User | 30 |
| 4.4 - Création du schéma de la base Stock..... | 31 |
| 4.5 - Insérer les données initiales de la base Stock | 31 |
| 4.6 - Création du schéma de la base Gestion | 31 |
| 4.7 - Insérer les données initiales de la base Gestion..... | 31 |
| 4.8 - Configuration de PuTTY..... | 31 |
| 4.9 - Déploiement de config-server | 35 |
| 4.9.1 - Installation de Git..... | 35 |
| 4.9.2 - Récupération de l'exécutable..... | 36 |
| 4.9.3 - Variables d'environnement..... | 36 |
| 4.9.4 - Lancer le microservice..... | 36 |
| 4.9.5 - Vérifications..... | 36 |
| 4.10 - Déploiement de user-api | 37 |
| 4.10.1 - Installation de Git..... | 37 |
| 4.10.2 - Récupération de l'exécutable | 37 |
| 4.10.3 - Variables d'environnement..... | 37 |
| 4.10.4 - Configuration | 38 |
| 4.10.5 - Lancer le microservice | 39 |
| 4.10.6 - Vérifications | 39 |
| 4.11 - Déploiement de stock-api..... | 40 |
| 4.11.1 - Installation de Git..... | 40 |
| 4.11.2 - Récupération de l'exécutable | 40 |
| 4.11.3 - Variables d'environnement..... | 40 |
| 4.11.4 - Configuration | 41 |
| 4.11.5 - Lancer le microservice | 42 |
| 4.11.6 - Vérifications | 42 |
| 4.12 - Déploiement des fichiers statiques..... | 42 |
| 4.12.1 - Installation de Git..... | 43 |
| 4.12.2 - Récupération des fichiers..... | 43 |
| 4.13 - Déploiement de web-api | 43 |
| 4.13.1 - Installation de Git..... | 43 |
| 4.13.2 - Récupération de l'exécutable | 43 |
| 4.13.3 - Variables d'environnement..... | 44 |
| 4.13.4 - Configuration | 44 |
| 4.13.5 - Lancer le microservice | 45 |
| 4.13.6 - Vérifications | 45 |
| 4.14 - Déploiement de production-api | 45 |
| 4.14.1 - Installation de Git..... | 46 |
| 4.14.2 - Récupération de l'exécutable | 46 |
| 4.14.3 - Variables d'environnement..... | 46 |
| 4.14.4 - Configuration | 46 |

| | |
|---|-----------|
| 4.14.5 - Lancer le microservice | 47 |
| 4.14.6 - Vérifications | 48 |
| 4.15 - Déploiement de gestion-api..... | 48 |
| 4.15.1 - Installation de Git..... | 48 |
| 4.15.2 - Récupération de l'exécutable | 48 |
| 4.15.3 - Variables d'environnement..... | 48 |
| 4.15.4 - Configuration | 49 |
| 4.15.5 - Lancer le microservice | 50 |
| 4.15.6 - Vérifications | 51 |
| 5 - Procédure de démarrage / arrêt | 52 |
| 5.1 - Base de données User | 52 |
| 5.1.1 - Préalable..... | 52 |
| 5.1.2 - Démarrage..... | 53 |
| 5.1.3 - Arrêt..... | 54 |
| 5.2 - Base de données Stock | 54 |
| 5.3 - Base de données Gestion..... | 54 |
| 5.4 - Config-server | 54 |
| 5.4.1 - Préalable..... | 54 |
| 5.4.2 - Démarrage..... | 54 |
| 5.4.3 - Arrêt..... | 55 |
| 5.5 - user-api | 55 |
| 5.5.1 - Préalable..... | 55 |
| 5.5.2 - Démarrage..... | 55 |
| 5.5.3 - Arrêt..... | 55 |
| 5.6 - stock-api | 55 |
| 5.6.1 - Préalable..... | 55 |
| 5.6.2 - Démarrage..... | 56 |
| 5.6.3 - Arrêt..... | 56 |
| 5.7 - web-api | 56 |
| 5.7.1 - Préalable..... | 56 |
| 5.7.2 - Démarrage..... | 56 |
| 5.7.3 - Arrêt..... | 56 |
| 5.8 - production-api | 56 |
| 5.8.1 - Préalable..... | 56 |
| 5.8.2 - Démarrage..... | 57 |
| 5.8.3 - Arrêt..... | 57 |
| 5.9 - gestion-api..... | 57 |
| 5.9.1 - Préalable..... | 57 |
| 5.9.2 - Démarrage..... | 57 |
| 5.9.3 - Arrêt..... | 57 |
| 6 - Procédure de mise à jour | 58 |
| 6.1 - Base de données User | 58 |
| 6.1.1 - Préalable..... | 58 |
| 6.1.2 - Mise à jour..... | 58 |

| | |
|--|-----------|
| 6.1.3 - Finalisation | 58 |
| 6.2 - Base de données Stock | 58 |
| 6.2.1 - Préalable..... | 58 |
| 6.2.2 - Mise à jour..... | 59 |
| 6.2.3 - Finalisation | 59 |
| 6.3 - Base de données Gestion..... | 59 |
| 6.3.1 - Préalable..... | 59 |
| 6.3.2 - Mise à jour..... | 59 |
| 6.3.3 - Finalisation | 60 |
| 6.4 - Microservice web-api..... | 60 |
| 6.4.1 - Préalable..... | 60 |
| 6.4.2 - Mise à jour..... | 60 |
| 6.4.3 - Finalisation | 60 |
| 6.5 - Microservice production-api..... | 60 |
| 6.5.1 - Préalable..... | 60 |
| 6.5.2 - Mise à jour..... | 61 |
| 6.5.3 - Finalisation | 61 |
| 6.6 - Microservicegestion-api | 61 |
| 6.6.1 - Préalable..... | 61 |
| 6.6.2 - Mise à jour..... | 61 |
| 6.6.3 - Finalisation | 61 |
| 6.7 - Serveur de configuration..... | 62 |
| 6.7.1 - Préalable..... | 62 |
| 6.7.2 - Mise à jour..... | 62 |
| 6.7.3 - Finalisation | 62 |
| 7 - Supervision/Monitoring | 63 |
| 7.1 - Supervision de l'application web..... | 63 |
| 8 - Procédure de sauvegarde et restauration | 67 |
| 8.1 - Base de données..... | 67 |
| 8.1.1 - Sauvegarde d'une base de données..... | 67 |
| 8.1.2 - Restauration d'une base de données..... | 69 |
| 8.2 - Microservices | 70 |
| 8.2.1 - Sauvegarde ancienne configuration..... | 70 |
| 8.2.2 - Restauration ancienne configuration | 70 |
| 9 - Glossaire | 71 |

1 - VERSIONS

| Auteur | Date | Description | Version |
|--------|------------|----------------------|---------|
| PEDSF | 09/05/2020 | Création du document | 1.0 |

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

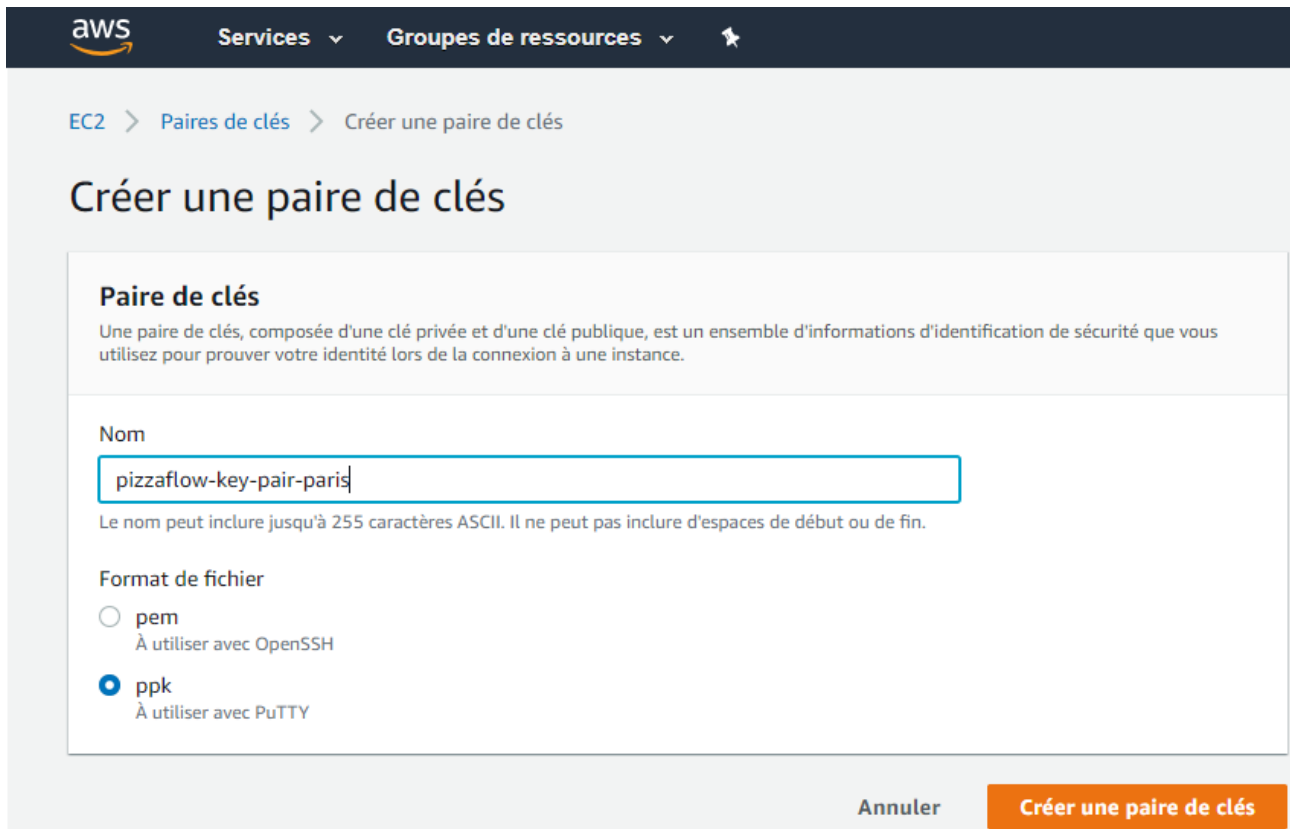
Pour de plus amples informations, se référer également aux éléments suivants :

- **DCT - PDOCPizza_01_fonctionnelle** : Dossier de conception fonctionnelle de l'application.
- **DCT - PDOCPizza_02_technique** : Dossier de conception technique de l'application.

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.



aws Services ▾ Groupes de ressources ▾

EC2 > Paires de clés > Créer une paire de clés

Créer une paire de clés

Paire de clés
Une paire de clés, composée d'une clé privée et d'une clé publique, est un ensemble d'informations d'identification de sécurité que vous utilisez pour prouver votre identité lors de la connexion à une instance.

Nom

Le nom peut inclure jusqu'à 255 caractères ASCII. Il ne peut pas inclure d'espaces de début ou de fin.

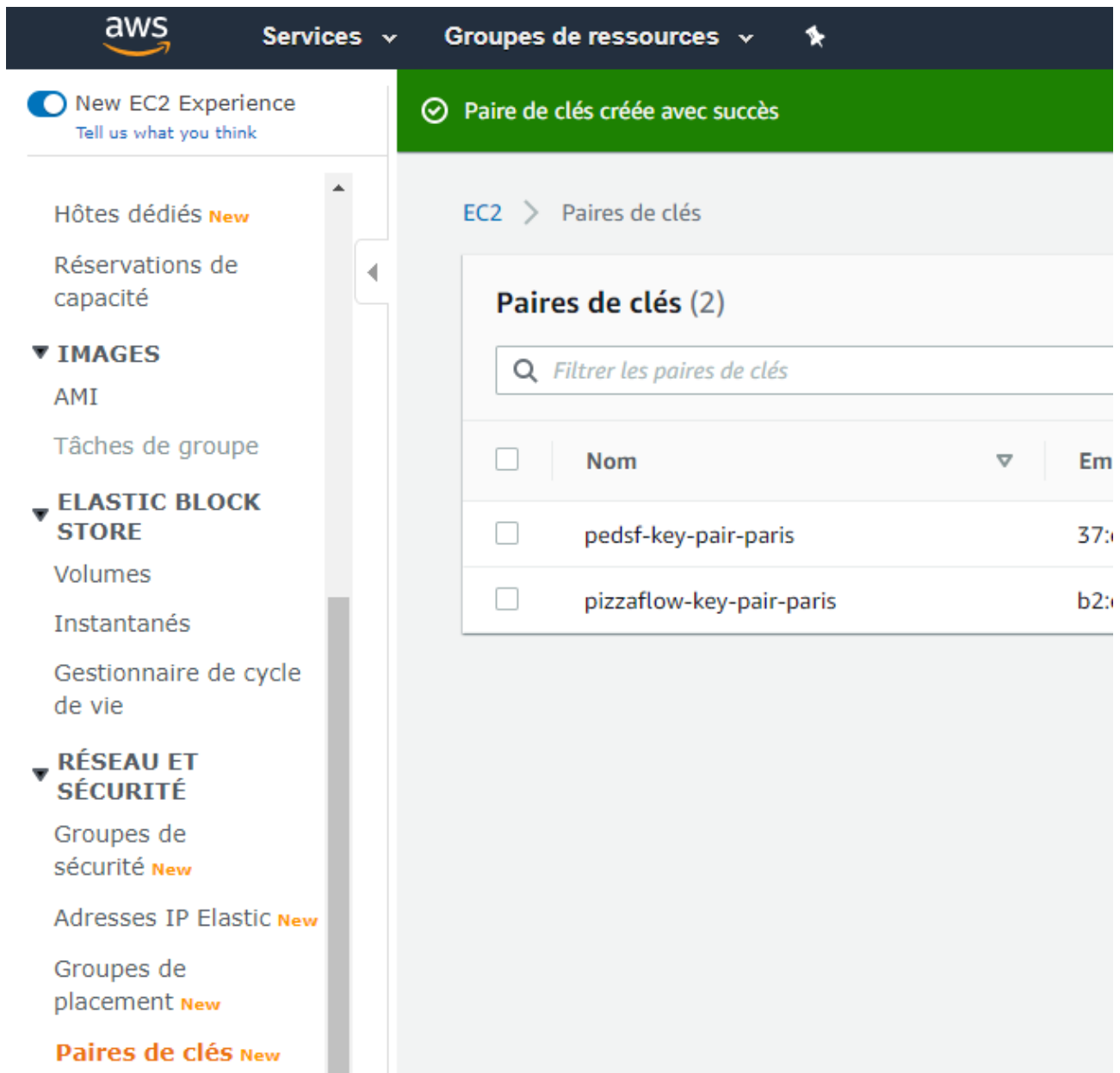
Format de fichier

☐ pem
À utiliser avec OpenSSH

☒ ppk
À utiliser avec PuTTY

Annuler Créer une paire de clés

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

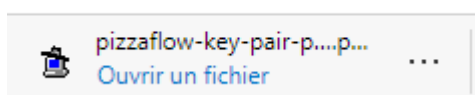
EC2 > Paires de clés

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 | Reserved hourly cost (1YR, No upfront) | vCPUs |
|-----------------------|--|-------|
| 0.412 USD | 0.2855 USD | 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 X

db.m5.large

| | | |
|-----------------------|--|-------|
| On-Demand hourly cost | Reserved hourly cost (1YR, No upfront) | vCPUs |
| 0.412 USD | 0.2855 USD | 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

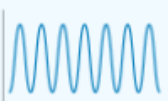
1 GiB

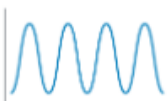
Any Network Perf...

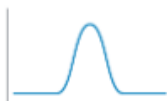
Workload [Informations](#)

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.

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

3.2.6 - 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.

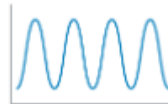
☐ Utilisation
constante



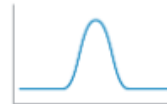
☒ Pic de trafic
quotidien



☐ 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

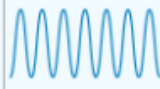
EC2 Instances (108)

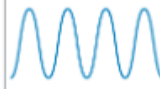
Selected Instance: c5.xlarge

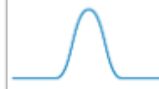
Workload [Informations](#)

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.

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

3.2.8 - 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

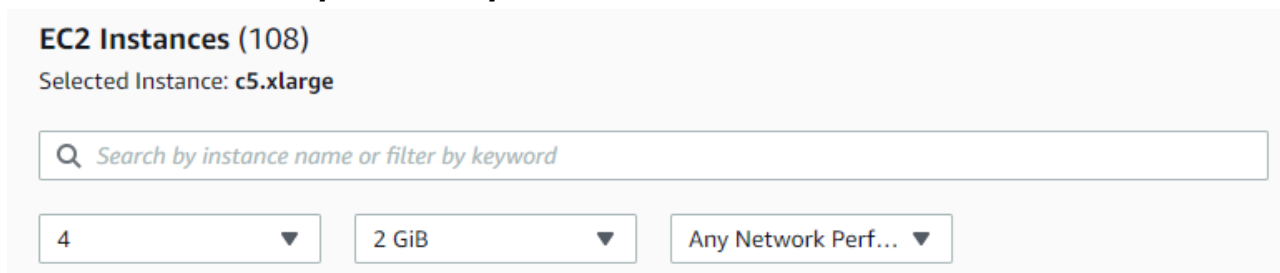
0

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 statics 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

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

...

3.3 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour dans le repository sur GitHub :

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

| BD | Schéma | Initial datas | Version |
|----------------|---------------------------------------|-------------------------------------|---------|
| user | /db/schema/pizzaflow.user.init.sql | /db/data/pizzaflow.user.data.sql | 1.0 |
| stock | /db/schema/pizzaflow.stock.init.sql | /db/data/pizzaflow.stock.data.sql | 1.0 |
| gestion | /db/schema/pizzaflow.gestion.init.sql | /db/data/pizzaflow.gestion.data.sql | 1.0 |

3.4 - Les contrôleurs

Les contrôleurs suivants doivent être accessibles et à jour dans le repository sur GitHub :

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

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

| Microservice | Fichier .jar | Version |
|--------------------|---------------------------------------|---------|
| user-api | /user-api/prod/1.0/user-api.jar | 1.0 |
| stock-api | /stock-api/prod/1.0/stock-api.jar | 1.0 |
| gestion-api | /gestion-api/prod/1.0/gestion-api.jar | 1.0 |

3.5 - Web-services

Les web services suivants doivent être accessibles et à jour dans le repository sur GitHub :

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

| Microservice | Fichier .jar | Version |
|-----------------------|---|---------|
| web-api | /web-api/prod/1.0/web-api.jar | 1.0 |
| production-api | /production-api/prod/1.0/production-api.jar | 1.0 |
| gestion-api | /gestion-api/prod/1.0/gestion-api.jar | 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 sur GitHub :

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

| Composant | Path | Version |
|----------------------------|---|---------|
| fichier JAR | /config-server/prod/1.0/config-server.jar | 1.0 |
| fichiers properties | /properties/prod/1.0/ | 1.0 |

3.7 - Les fichiers statiques

Les fichiers statiques pour les applications web doivent être accessibles et à jour dans le repository sur GitHub :

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

| Répertoire | Path | Version |
|------------------|---------------------------|---------|
| images | /resources/static/images/ | 1.0 |
| templates | /resources/templates/ | 1.0 |
| locales | /resources/locale/ | 1.0 |
| css | /resources/static/css/ | 1.0 |
| js | /resources/static/js/ | 1.0 |

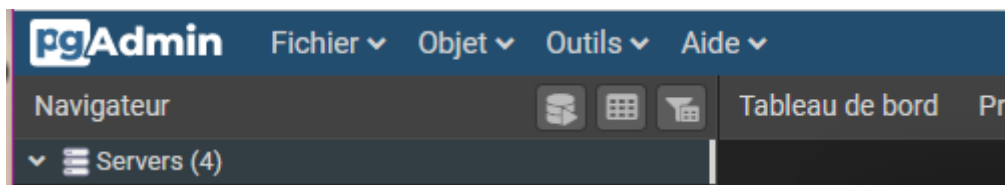
4 - PROCÉDURE DE DÉPLOIEMENT

4.1 - 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 :

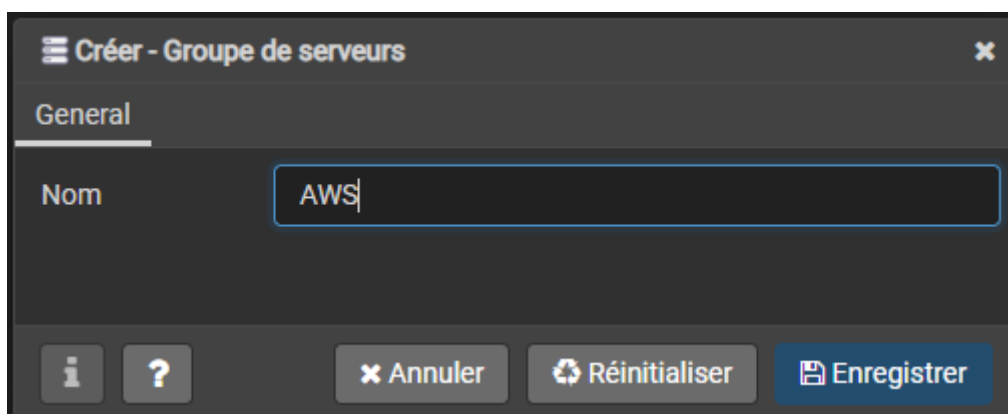
<https://www.pgadmin.org/download/>

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

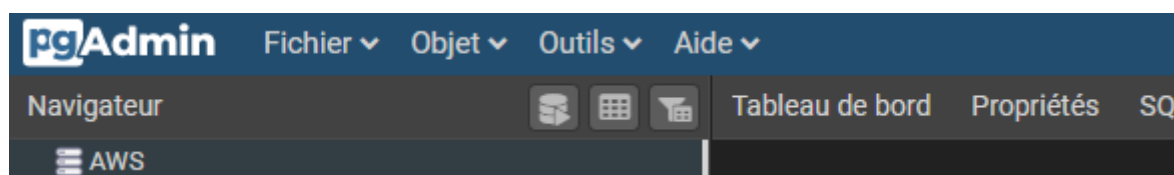


Il faut créer un groupe de serveur pour classer les connexions aux bases de données dans le menu :

Objet/Créer/Groupe de serveur



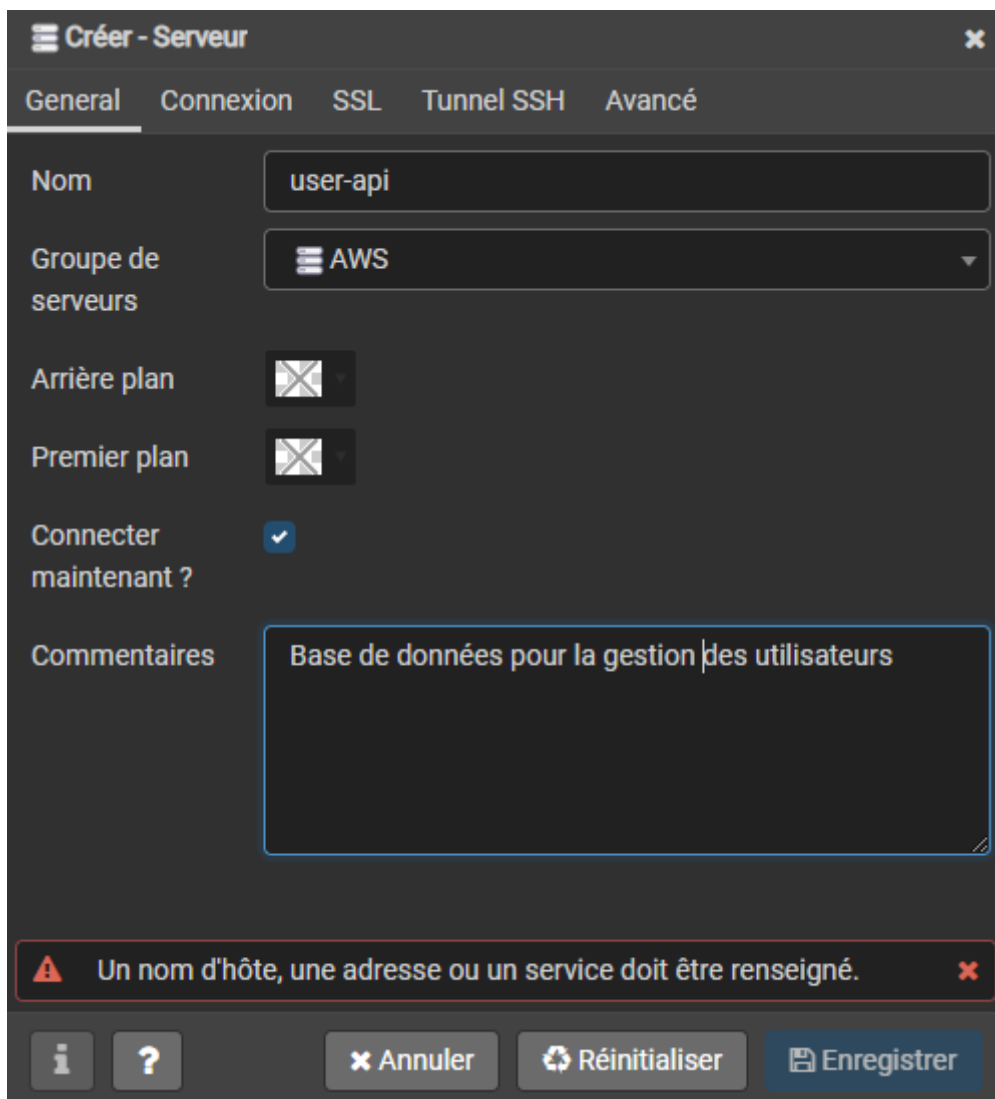
Sélectionner le groupe de serveur créé.



Cliquer sur le menu :

Objet/Créer/Serveur

Indiquer le "Nom" de la connexion et une description dans la partie "Commentaires".



Créer - Serveur

General Connexion SSL Tunnel SSH Avancé

Nom: user-api

Groupe de serveurs: AWS

Arrière plan: ☐

Premier plan: ☐

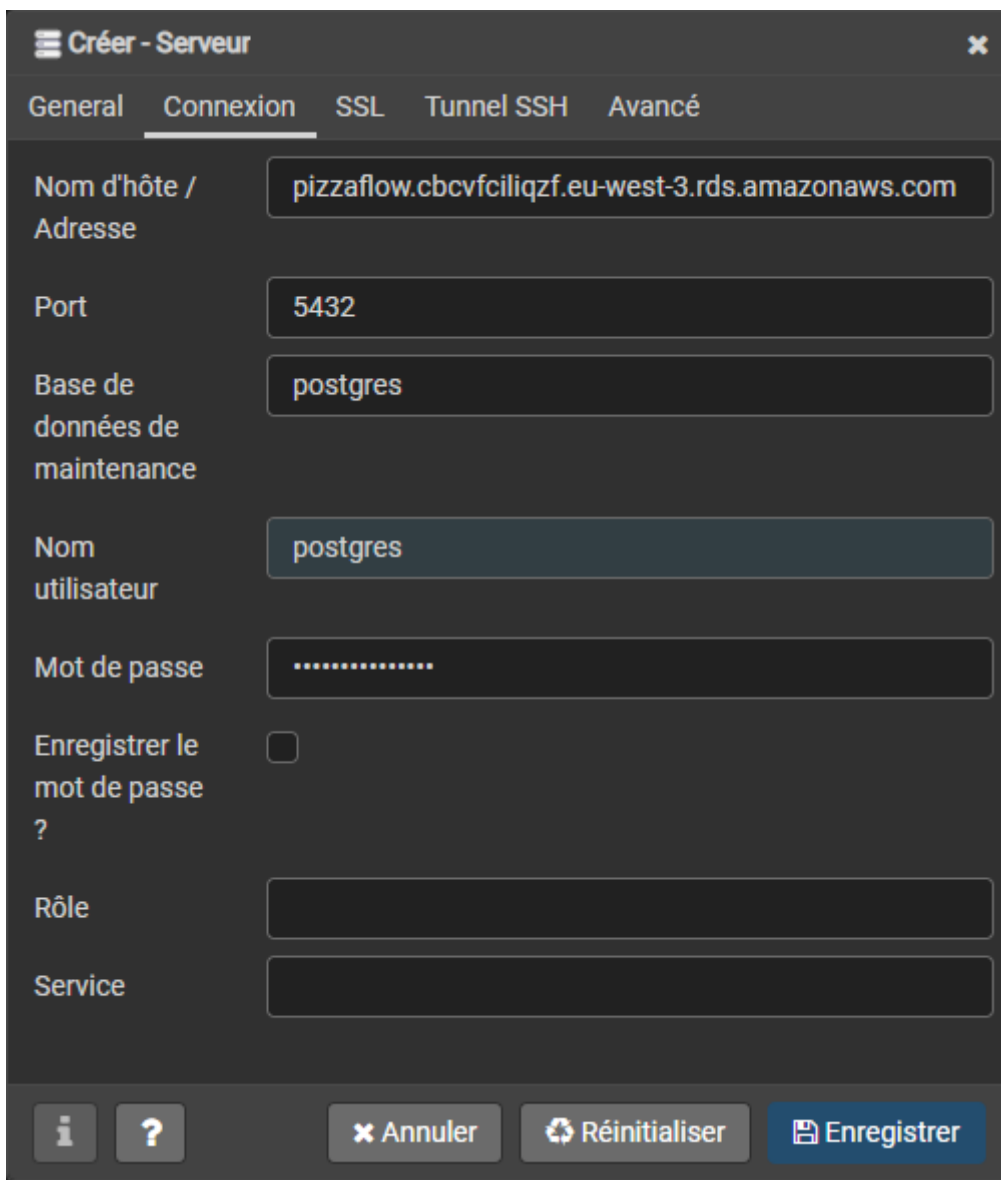
Connecter maintenant ? ☒

Commentaires: Base de données pour la gestion des utilisateurs

Un nom d'hôte, une adresse ou un service doit être renseigné.

Annuler Réinitialiser Enregistrer

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.



Il faut créer une connexion pour chaque base de données.

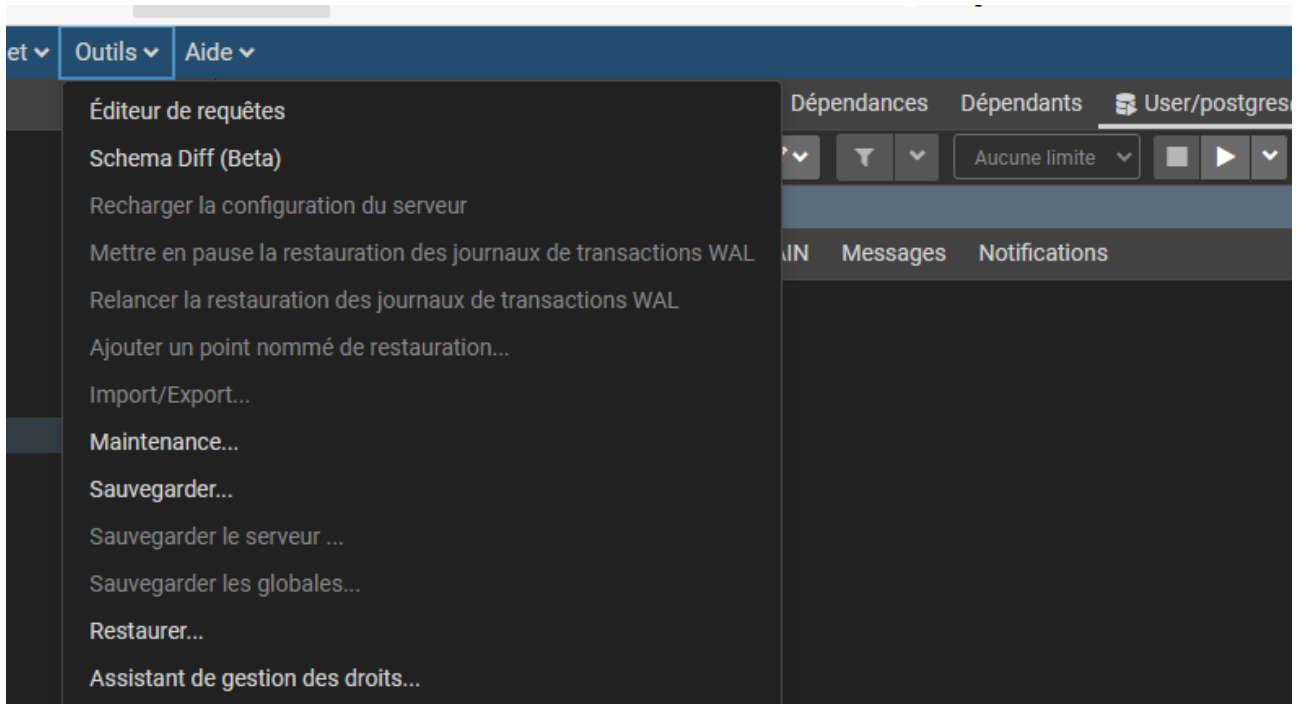
4.2 - Création du schéma de la base User

Récupérer le schéma de la base de données **User** sur le repository **Git**.

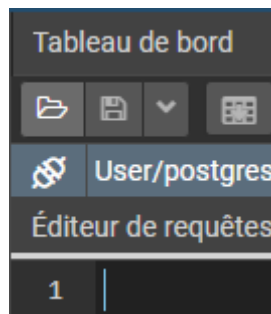
<https://github.com/pizzaflow/db/schema/pizzaflow.user.init.sql>

Dans **pgAdmin** il faut sélectionner la base de données **User** et le menu **Outils/Editeur de requêtes**.

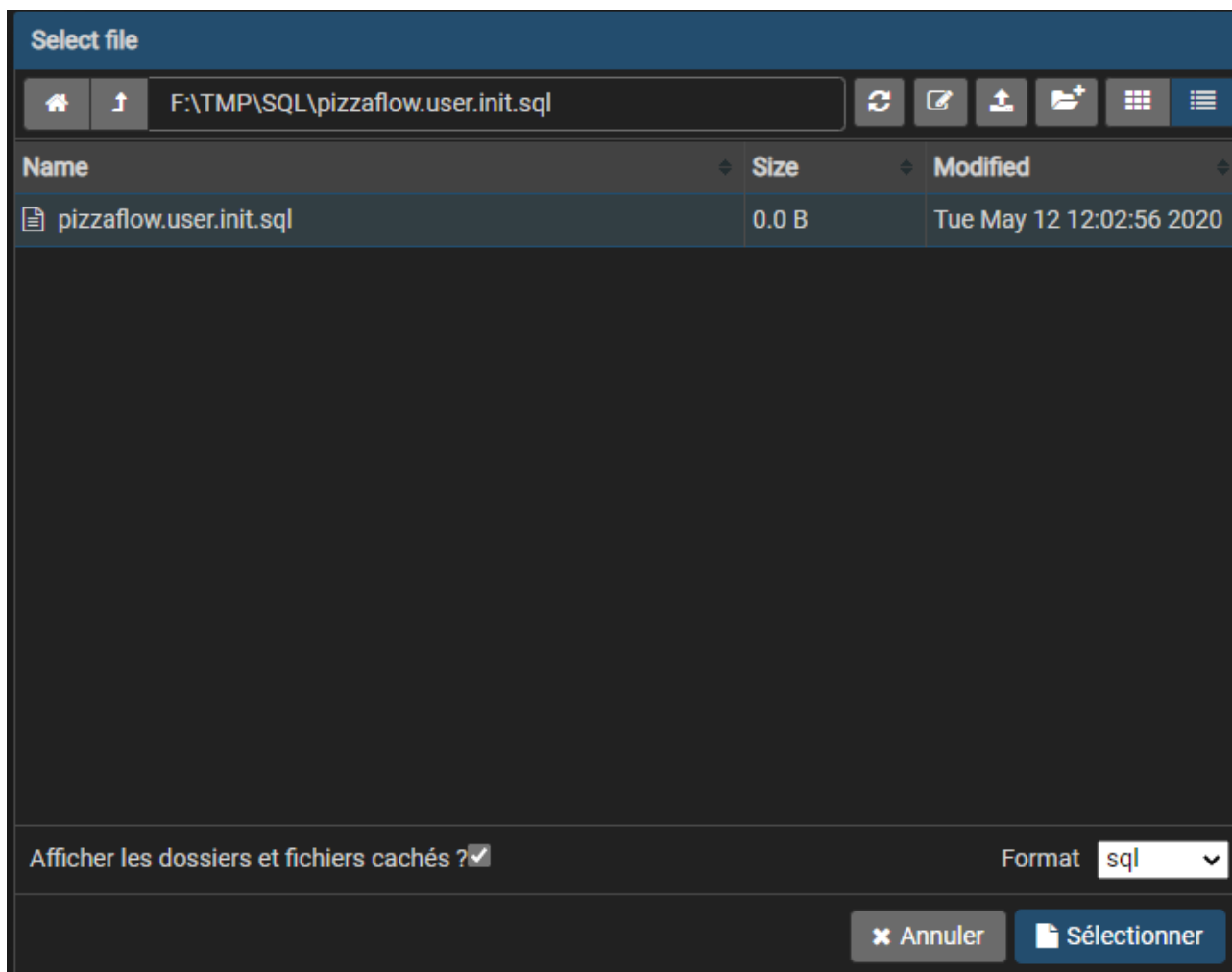
La fenêtre de l'éditeur s'ouvre dans la partie gauche de l'écran.



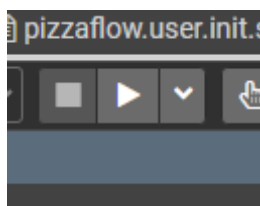
Cliquer sur l'icône "Ouvrir fichier" en haut à gauche de la fenêtre de requêtes.



Sélectionner le fichier **pizzaflow.user.init.sql** et valider.



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



4.3 - Insérer les données initiales de la base User

Récupérer les données initiales de la base de données **User** sur le repository **Git**.

<https://github.com/pizzaflow/db/schema/pizzaflow.user.data.sql>

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

4.4 - Création du schéma de la base Stock

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

<https://github.com/pizzaflow/db/schema/pizzaflow.stock.init.sql>

4.5 - Insérer les données initiales de la base Stock

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

<https://github.com/pizzaflow/db/schema/pizzaflow.stock.data.sql>

4.6 - Création du schéma de la base Gestion

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

<https://github.com/pizzaflow/db/schema/pizzaflow.gestion.init.sql>

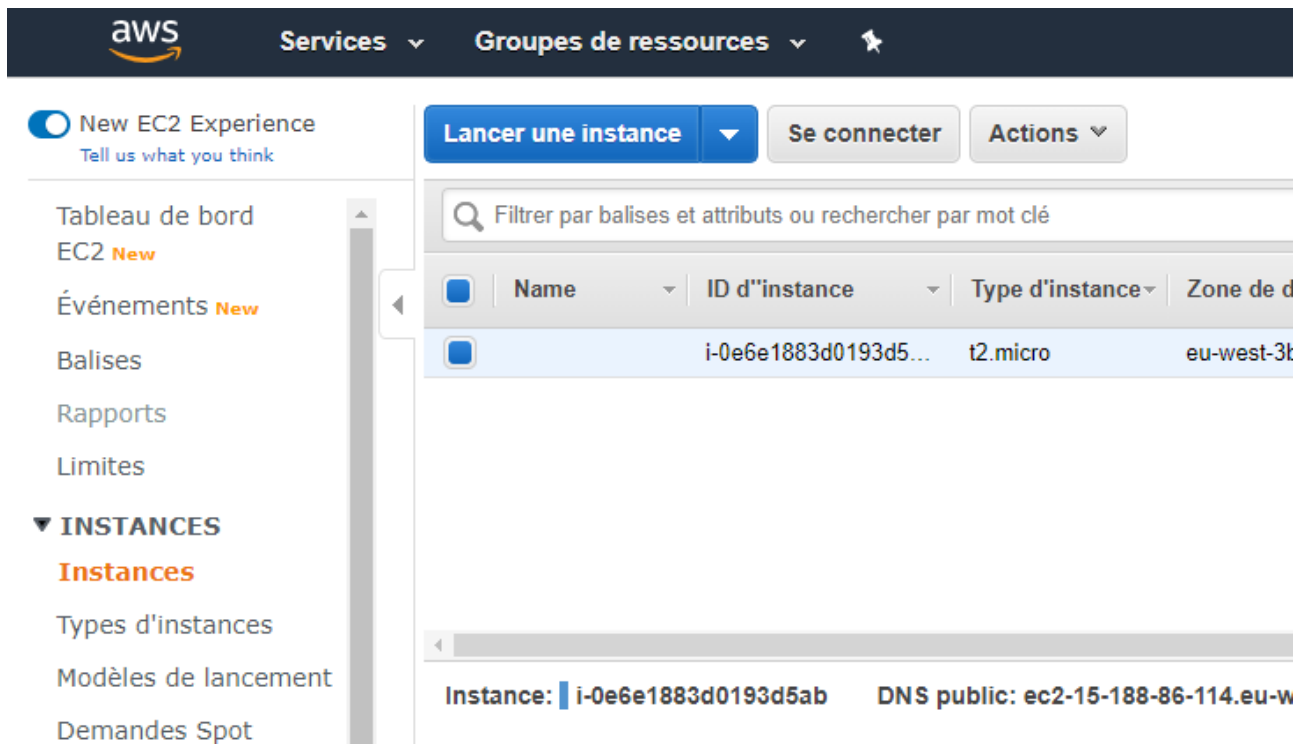
4.7 - Insérer les données initiales de la base Gestion

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

<https://github.com/pizzaflow/db/schema/pizzaflow.gestion.data.sql>

4.8 - 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".



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

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 d |
|--------------------------|--------|----------------------|-------------------|------------|
| <input type="checkbox"/> | | i-0e6e1883d0193d5... | t2.micro | eu-west-3k |

Instance: **i-0e6e1883d0193d5ab** 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 :

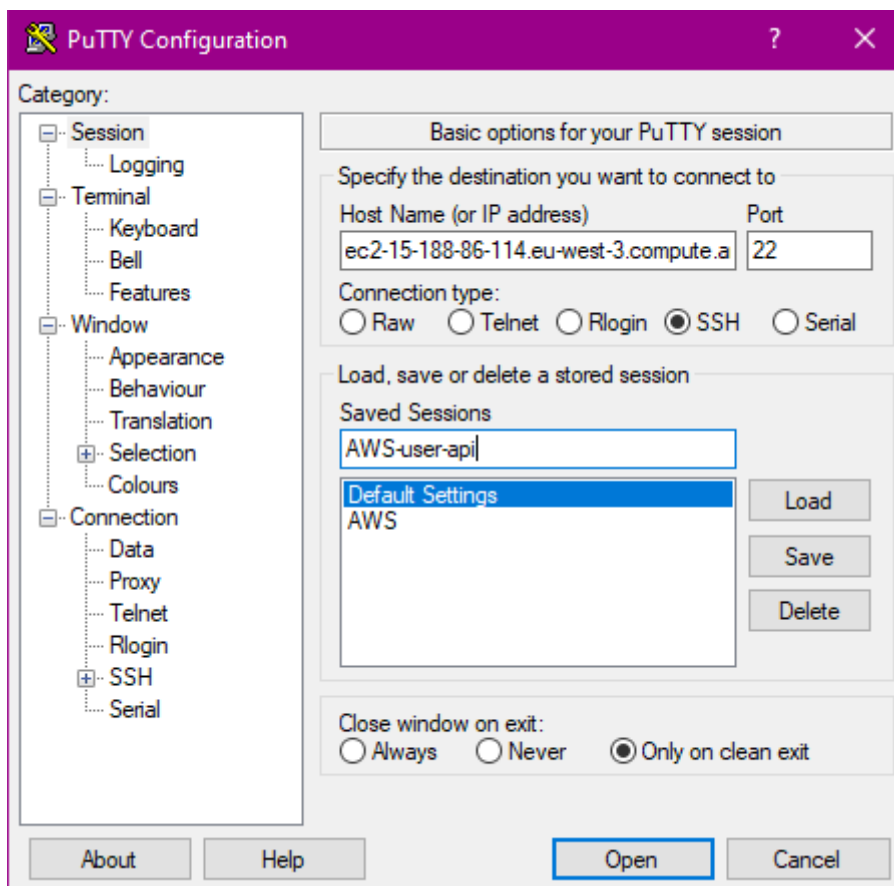
```
ssh -i "pizzaflow-key-pair-paris.pem" ec2-user@ec2-15-236-51-101.eu-west-3.compute.amazonaws.com
```

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.

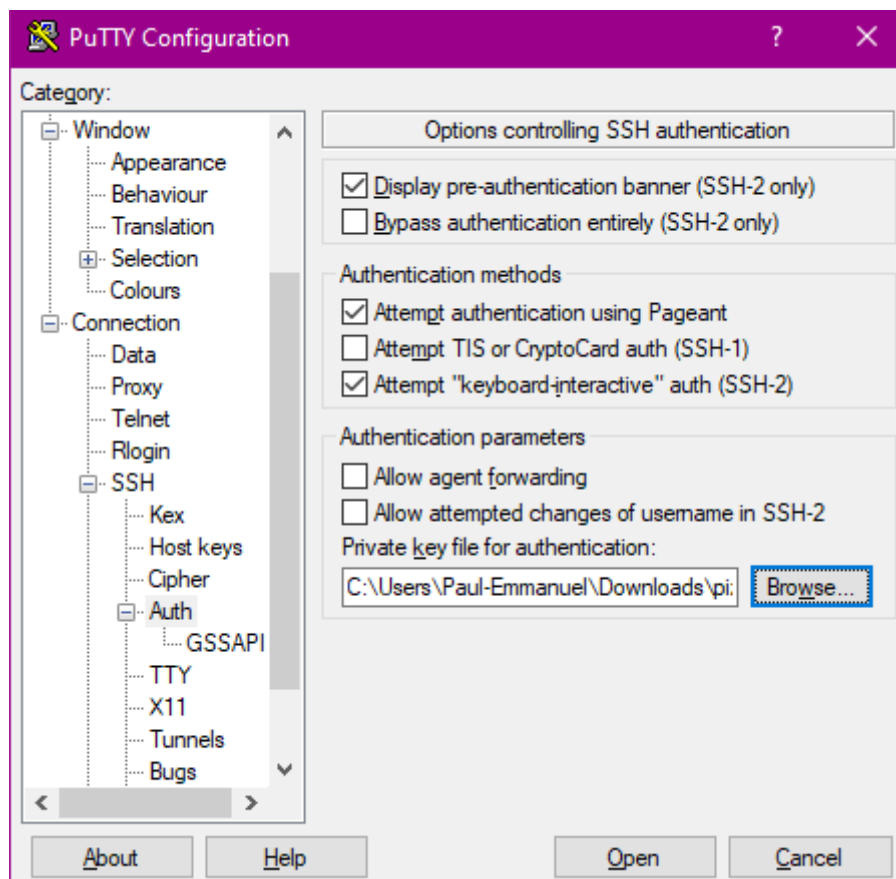
Si vous avez besoin d'aide pour vous connecter à votre instance, veuillez consulter notre [documentation de connexion](#).

[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 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.

4.9 - Déploiement de config-server

On se connecte à la **VM** du edge microservice avec **PuTTY**.

4.9.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

sudo yum update -y

sudo yum install git -y

4.9.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/config-server
```

4.9.3 - 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 **start.sh**.

| Nom | Obligatoire | Description |
|-----------------|-------------|--|
| JASYPT_SECRET | oui | Clé pour de cryptage des données sensibles des fichiers de propriétés. |
| PROPERTIES_PATH | oui | Chemin du repository contenant les fichiers de propriétés |

4.9.4 - 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/config-server
```

```
sudo chmod g+x u+x *.sh
```

```
sudo ./start.sh
```

4.9.5 - Vérifications

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

```
ps -ef | grep config-server
```

4.10 - Déploiement de user-api

On se connecte à la **VM** du microservice avec **PuTTY**.

4.10.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.10.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/user-api
```

4.10.3 - 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 **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_DB_USERNAME | oui | Identifiant de connexion à la base de données User |
| USER_DB_PASSWORD | oui | Mot de passe correspondant |

4.10.4 - 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

## 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/user-api.log
```

4.10.5 - 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/user-api
sudo chmod g+x u+x *.sh
sudo ./start.sh
```

4.10.6 - Vérifications

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

```
ps -ef | grep user-api
```

4.11 - Déploiement de stock-api

On se connecte à la **VM** du microservice avec **PuTTY**.

4.11.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.11.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/stock-api
```

4.11.3 - 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 **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 |
| STOCK_DB_USERNAME | oui | Identifiant de connexion à la base de données Stock |
| STOCK_DB_PASSWORD | oui | Mot de passe correspondant |

4.11.4 - 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 vrai 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
```

4.11.5 - 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
sudo chmod g+x u+x *.sh
sudo ./start.sh
```

4.11.6 - Vérifications

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

```
ps -ef | grep stock-api
```

4.12 - Déploiement des fichiers statiques

Se connecter au bucket **S3** avec **PuTTY** pour faire l'installation.

4.12.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.12.2 - Récupération des fichiers

Se placer à la racine est faire un **git clone**.

```
cd /  
sudo git clone https://github.com/pizzaflow/resources.git
```

4.13 - Déploiement de web-api

On se connecte à la **VM** du microservice avec **PuTTY**.

4.13.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.13.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/web-api
```

4.13.3 - 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 **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 user-api |
| STOCK_API_IP | oui | Adresse IP de stock-api |
| RESOURCES_PATH | oui | Chemin vers le bucket des ressources |

4.13.4 - 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
```

4.13.5 - 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/web-api
sudo chmod g+x u+x *.sh
sudo ./start.sh
```

4.13.6 - Vérifications

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

```
ps -ef | grep web-api
```

4.14 - Déploiement de production-api

On se connecte à la **VM** du microservice avec **PuTTY**.

4.14.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.14.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/production-api
```

4.14.3 - 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 **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 user-api |
| STOCK_API_IP | oui | Adresse IP de stock-api |
| WEB_API_IP | oui | Adresse IP de web-api |
| RESOURCES_PATH | oui | Chemin vers le bucket des ressources |

4.14.4 - Configuration

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

| | |
|----------------------------|--|
| 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 |
|----------------------------|--|

<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
```

4.14.5 - 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  
sudo chmod g+x u+x *.sh  
sudo ./start.sh
```

4.14.6 - Vérifications

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

```
ps -ef | grep production-api
```

4.15 - Déploiement de gestion-api

On se connecte à la **VM** du microservice avec **PuTTY**.

4.15.1 - Installation de Git

Installer l'outil de versioning Git pour récupérer les fichiers avec les commandes suivante :

```
sudo yum update -y  
sudo yum install git -y
```

4.15.2 - Récupération de l'exécutable

Se placer dans le répertoire **/srv** de l'instance.

```
cd /srv
```

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git clone https://raw.githubusercontent.com/pizzaflow/master/gestion-api
```

4.15.3 - 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 **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 user-api |
| STOCK_API_IP | oui | Adresse IP de stock-api |
| WEB_API_IP | oui | Adresse IP de web-api |
| 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 |

4.15.4 - 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 vrai 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
```

4.15.5 - Lancer le microservice

Un script de commande fourni permet de lancer le microservice en tâche de fond après avoir défini

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

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  
sudo chmod g+x u+x *.sh  
sudo ./start.sh
```

4.15.6 - Vérifications

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

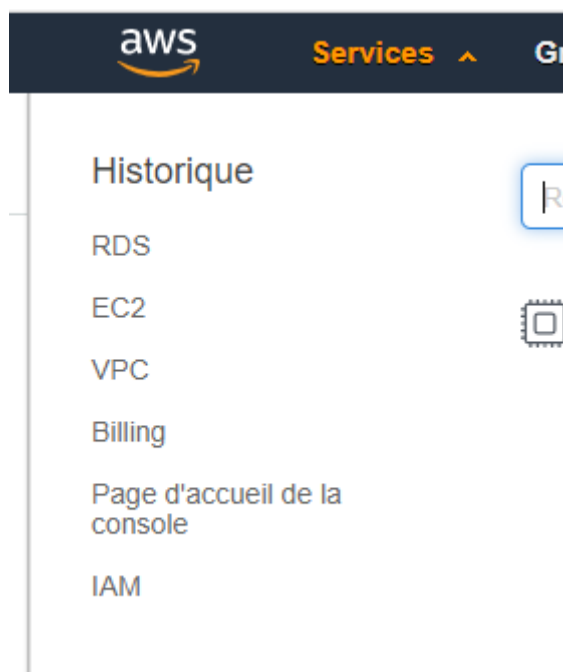
```
ps -ef | grep gestion-api
```

5 - PROCÉDURE DE DÉMARRAGE / ARRÊT

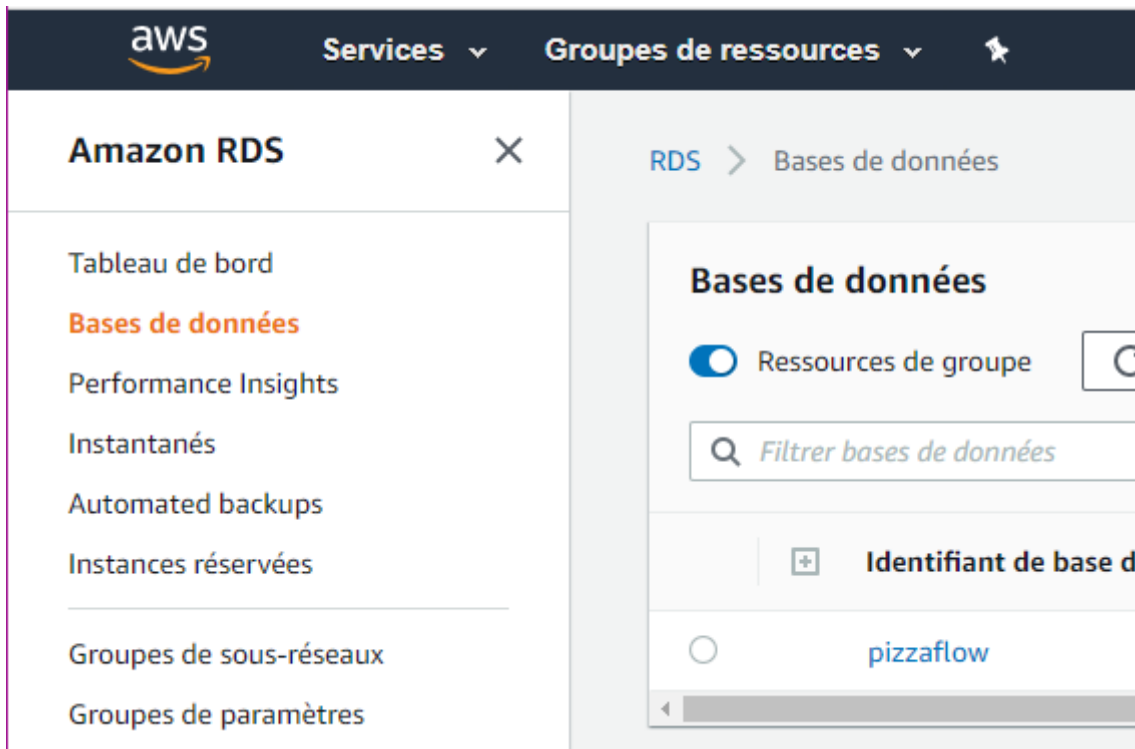
5.1 - Base de données User

5.1.1 - Préalable

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



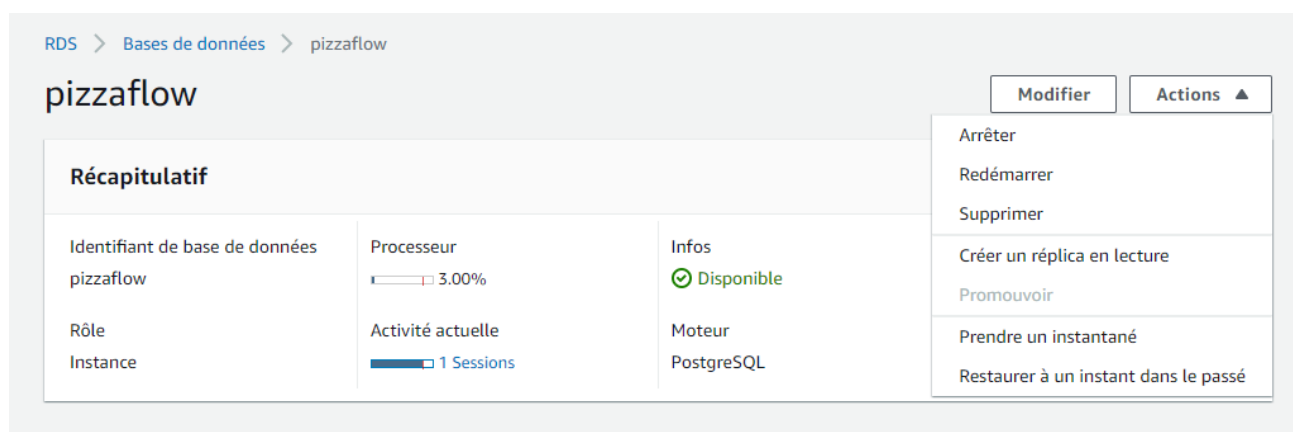
Choisir "Bases de données" pour voir les instances des bases de données.



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

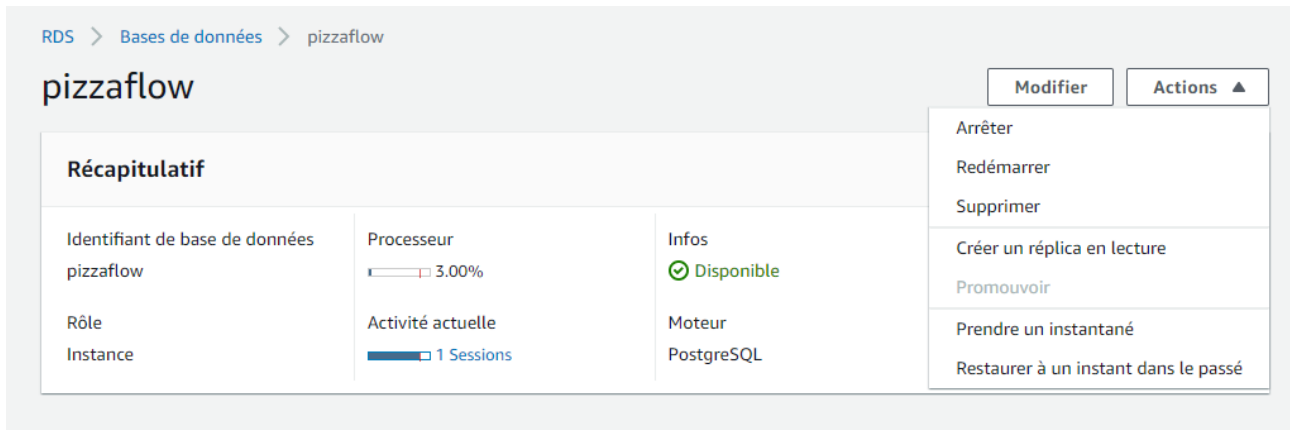
5.1.2 - Démarrage

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
- 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.

sudo ./start.sh

5.4.3 - Arrêt

Exécuter le script d'arrêt.

```
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.

```
sudo ./start.sh
```

5.5.3 - Arrêt

Exécuter le script d'arrêt.

```
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.

```
sudo ./start.sh
```

5.6.3 - Arrêt

Exécuter le script d'arrêt.

```
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.

```
sudo ./start.sh
```

5.7.3 - Arrêt

Exécuter le script d'arrêt.

```
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

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

l'application.

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

5.8.2 - Démarrage

Exécuter le script de démarrage.

```
sudo ./start.sh
```

5.8.3 - Arrêt

Exécuter le script d'arrêt.

```
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.

```
sudo ./start.sh
```

5.9.3 - Arrêt

Exécuter le script d'arrêt.

```
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 web-api

6.4.1 - Préalable

Aller dans le répertoire de l'application avec la commande suivante :

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

Arrêter le microservice avec la commande :

```
sudo ./shutdown.sh
```

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

6.4.2 - Mise à jour

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git pull https://raw.githubusercontent.com/pizzaflow/master/web-api
```

6.4.3 - Finalisation

Relancer le microservice **web-api**.

6.5 - Microservice production-api

6.5.1 - Préalable

Aller dans le répertoire de l'application avec la commande suivante :

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

Arrêter le microservice avec la commande :

```
sudo ./shutdown.sh
```

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

6.5.2 - Mise à jour

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git pull https://raw.githubusercontent.com/pizzaflow/master/production-api
```

6.5.3 - Finalisation

Relancer le microservice **production-api**.

6.6 - Microservicegestion-api

6.6.1 - Préalable

Aller dans le répertoire de l'application avec la commande suivante :

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

Arrêter le microservice avec la commande :

```
sudo ./shutdown.sh
```

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

6.6.2 - Mise à jour

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git pull https://raw.githubusercontent.com/pizzaflow/master/gestion-api
```

6.6.3 - Finalisation

Relancer le microservice **gestion-api**.

6.7 - Serveur de configuration

6.7.1 - Préalable

Aller dans le répertoire de l'application avec la commande suivante.

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

Arrêter le serveur avec la commande :

```
sudo ./shutdown.sh
```

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

6.7.2 - Mise à jour

Récupérer le **JAR** du microservice qui est dans un repository de **GitHub** avec la commande :

```
sudo git pull https://raw.githubusercontent.com/pizzaflow/master/config-server
```

6.7.3 - Finalisation

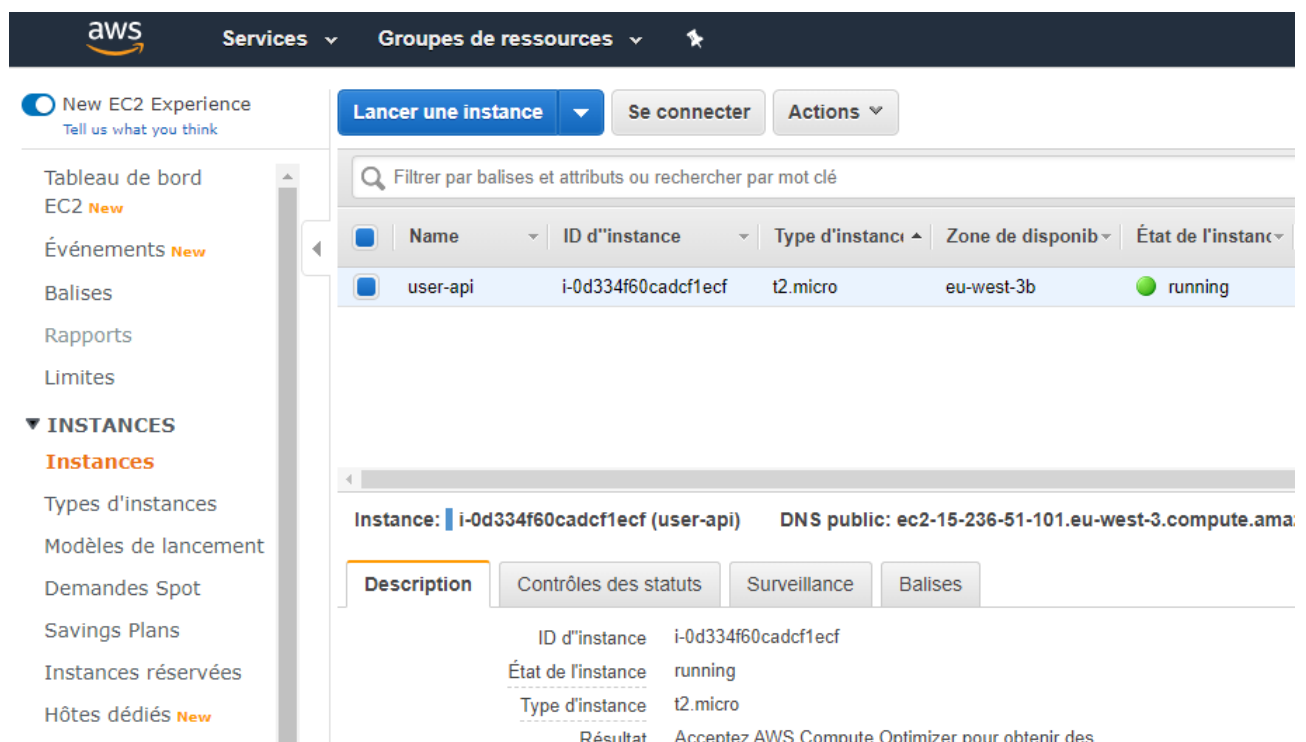
Relancer le microservice **config-server**.

7 - SUPERVISION/MONITORING

7.1 - Supervision de l'application web-api

Lancer un navigateur à l'adresse 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.



The screenshot shows the AWS Management Console interface. On the left, there is a navigation menu with options like 'Tableau de bord', 'EC2 New', 'Événements', 'Balises', 'Rapports', 'Limites', and 'INSTANCES'. The 'INSTANCES' section is expanded, showing 'Instances' as the selected option. The main area displays a table of instances. One instance, 'user-api', is listed with ID 'i-0d334f60cadcf1ecf', type 't2.micro', and state 'running'. Below the table, there are tabs for 'Description', 'Contrôles des statuts', 'Surveillance', and 'Balises'. The 'Description' tab is active, showing details for the selected instance: ID d'instance 'i-0d334f60cadcf1ecf', État de l'instance 'running', Type d'instance 't2.micro', and a result message '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.

Description

Contrôles des statuts

Surveillance

Balises

► **Alarmes CloudWatch :**  **Aucune alarme configurée**

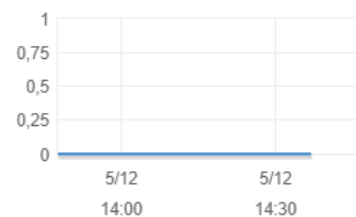
Métriques de CloudWatch : Surveillance basique. [Activer la surveillance détaillée](#)

Vous trouverez ci-dessous vos métriques CloudWatch pour les ressources sélectionnées (un maximum de 10). Cliquez sur un graphique [CloudWatch](#)

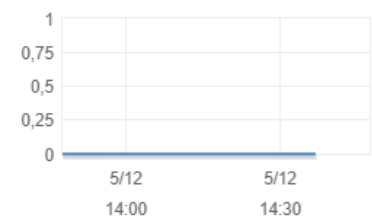
Utilisation de la CPU (Pourcentage)



Lectures sur disque (Octets)



Opérations de lecture sur disque (Transactions)



Réseau entrant (Octets)



Réseau sortant (Octets)



Paquets de réseau entrant (Nombre)



Échec du contrôle de statut (instance) (Nombre)

Échec du contrôle de statut (système) (Nombre)

Utilisation du crédit pour CPU (Nomb

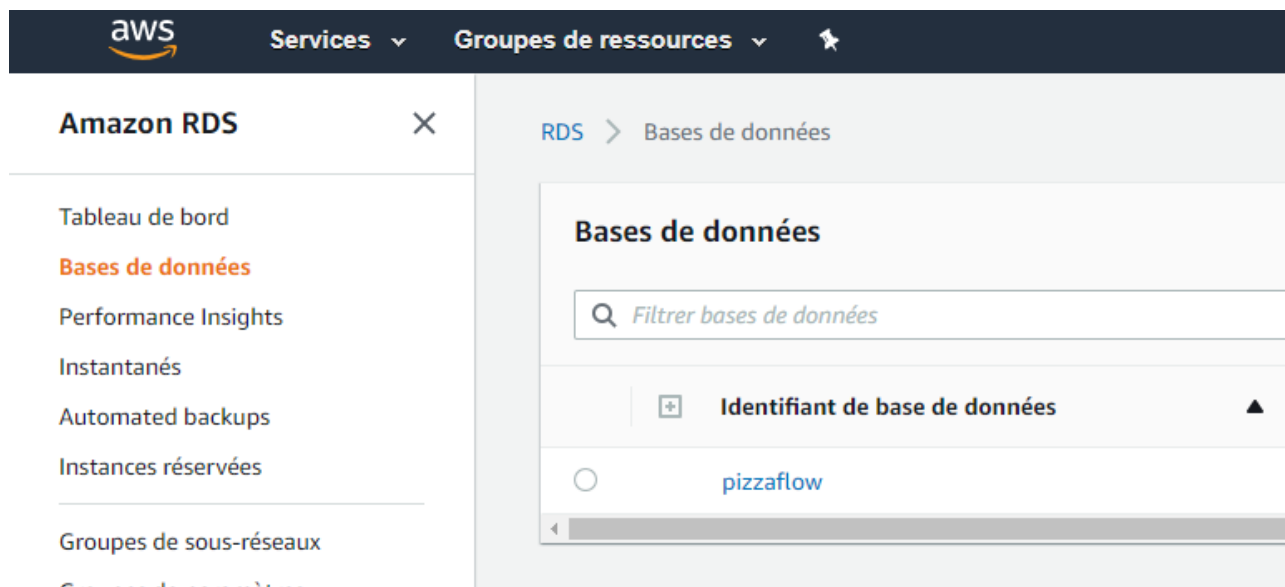
7.2 - Supervision des microservices

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

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 comme ci-dessus.

7.3 - Supervision 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 > Bases de données > 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)

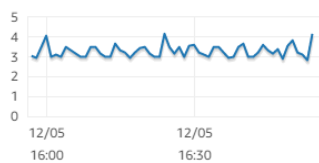


Ajouter une instance à c

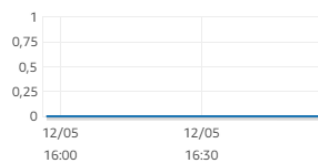
Légende : pizzaflow

Q

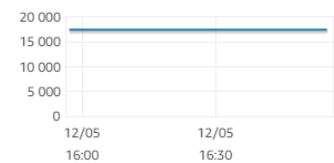
Utilisation de l'UC (Pourcentage)



Connections DB (Nombre)



Espace de stockage disponible (Mo)

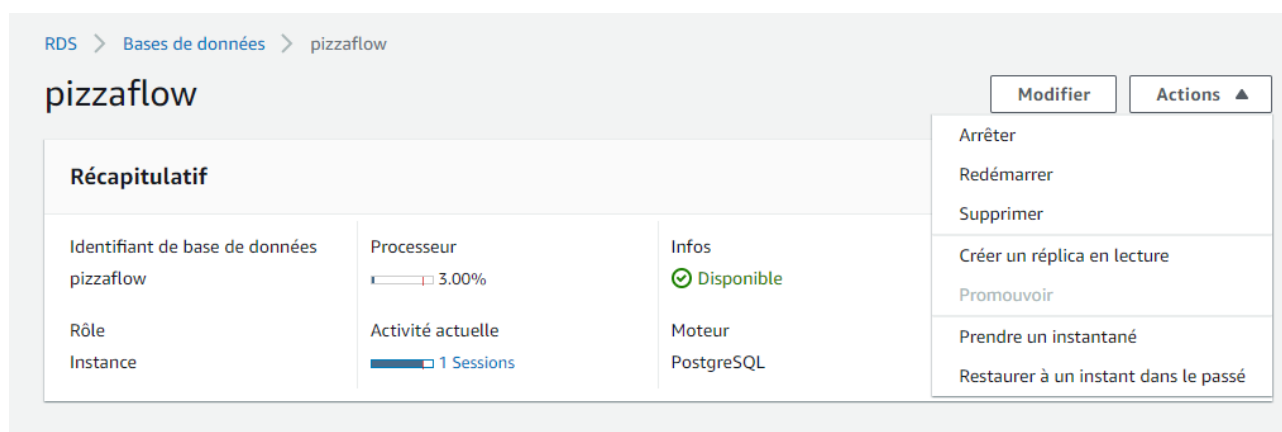


8 - PROCÉDURE DE SAUVEGARDE ET RESTAURATION

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".



RDS > Bases de données > pizzaflow

pizzaflow

Récapitulatif

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

Actions

- Arrêter
- Redémarrer
- Supprimer
- Créer un réplica en lecture
- Promouvoir
- Prendre un instantané**
- 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

aws

Services

Groupe 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

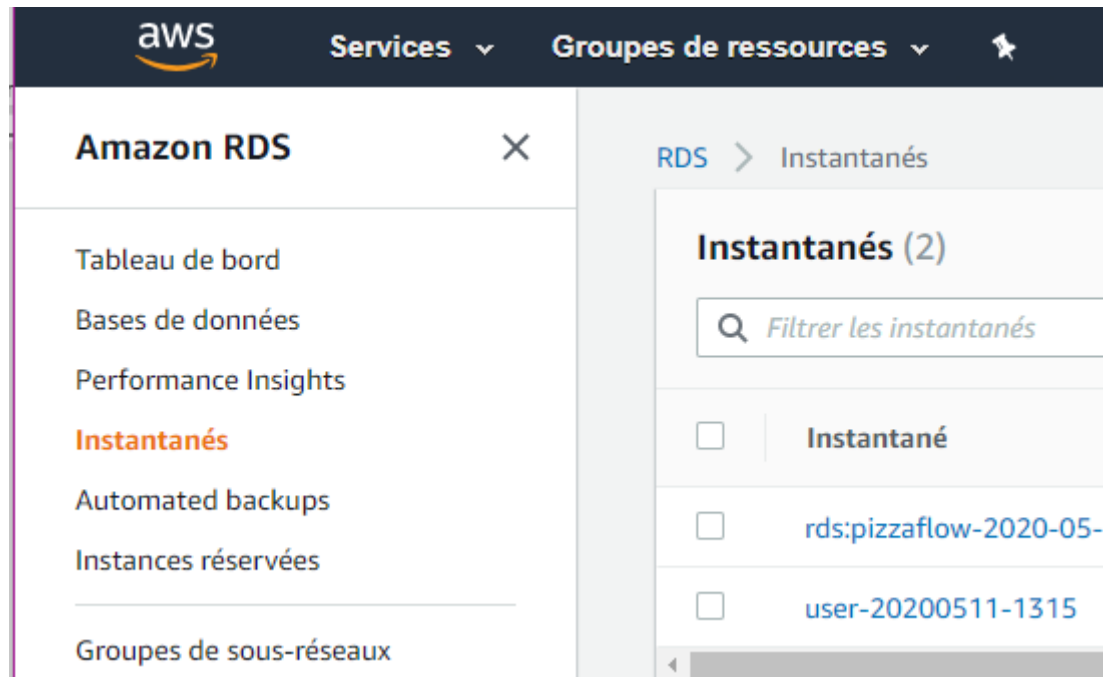
Il apparaît ensuite dans la liste des instantanés de la base.

| 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 |

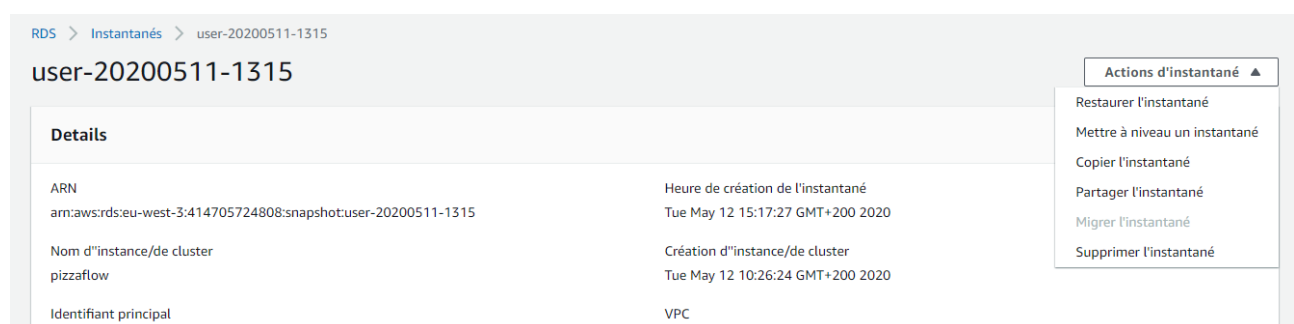
Il suffira de le sélectionner l'instantané pour restaurer la base.

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 ancienne configuration

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.

```
cd /srv/[nom-microservice]
```

Arrêter le microservice avec la commande :

```
sudo ./shutdown.sh
```

Lancer le script de sauvegarde du système avec la commande suivante :

```
sudo ./backup.sh
```

Le script créer un répertoire dont le nom est au format :

```
nom-microservice-YYYYMMDD-HHMM
```

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

```
sudo ./start.sh
```

8.2.2 - Restauration ancienne configuration

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.

```
cd /srv/[nom-microservice]
```

Arrêter le microservice avec la commande :

```
sudo ./shutdown.sh
```

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

```
sudo ./restore.sh YYYYMMDD-HHMM
```

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

```
sudo ./start.sh
```

9 - GLOSSAIRE

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