

Mis en place d'une plateforme de développement



Sommaire

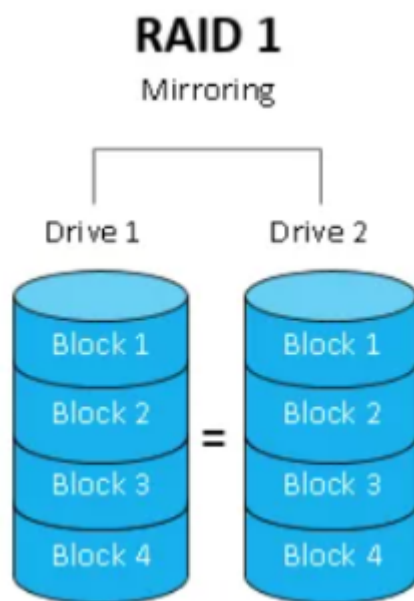
A quoi ressemble notre environnement de développement à distance ?	2
A quoi ressemble notre environnement de développement local ?	3
Pourquoi avoir choisi ces logiciels ?	3
Docker automatisation de la création de notre environnement	4
Lien entre nginx et php	6
Associer pgadmin4 et postgresql	10

A quoi ressemble notre environnement de développement à distance ?

Pour notre environnement de développement nous utilisons docker et 4 conteneurs¹ distinctes qui sont nginx php postgres et pgadmin.

Tout ceci est exécuté sur le NAS de Fatih l'avantage est que nous avons accès à distance à l'environnement de développement notamment grâce au reverse proxy qui nous renvoie directement aux pages web souhaitées :

- pour notre serveur web : <https://nginx.fidanhome.ovh>
- pour pgadmin : <https://pgadmin.fidanhome.ovh>



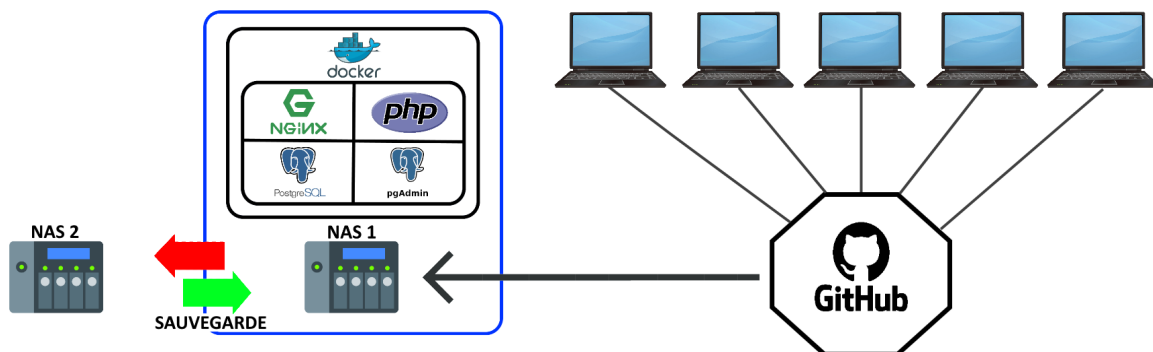
De plus, le Nas est en RAID 1 c'est-à-dire qu'il copie les données sur deux disques différents si l'un tombe en panne nos données sont stockées dans le second.

Et le Nas communique également avec un autre Nas avec lequel des sauvegardes de données sont effectuées tous les jours.

De plus, notre projet se trouve également sur github, nous envoyons donc nos fichiers sur github et Fatih s'occupe de mettre à jour les fichiers du nas avec git ce qui nous permet d'avoir régulièrement notre application web à jour sur le nas.

Le Nas et les conteneurs fonctionnent sous linux.

Vous pouvez retrouver un schéma récapitulatif du fonctionnement ci-dessous :



¹

A quoi ressemble notre environnement de développement local ?

Pour notre environnement de développement local nous avons utilisé docker afin d'avoir un serveur web local, nous utilisons par contre la même base de données qui est hébergée sur le nas.

De plus nous utilisons visual studio code pour éditer nos codes, plus particulièrement on utilise 2 extensions dans visual studio code qui est live-share qui va nous permettre de partager notre code à distance et le modifier en tant réelle un peu comme google doc, l'autre extensions est live-server pour le côté frontend de notre projet c'est une extension qui affiche en tant réelle la page web et les modification apportée dans le code.

Pourquoi avoir choisi ces logiciels ?

Docker : Docker est un outil de containerisation qui permet de faire tourner des images², l'avantage de docker est d'avoir des services séparés, si une image bug il suffira de le supprimer et de le recréer alors que dans une machine virtuel cela aurait pu tout casser.

Nginx : Nous avons choisi Nginx comme serveur web, car celui-ci est de plus en plus utilisé par les utilisateurs. De plus, selon le site de nginx : <https://www.nginx.com/resources/wiki/start/topics/examples/likeapache-htaccess/> celui-ci sera nettement plus performant que apache2.

PostgreSQL : Nous avons l'habitude de travailler avec notre équipe sur postgresql, même si on aurait pu utiliser d'autre service de base tel que MariaDB.

Pgadmin4 : Pgadmin4 nous permet de gérer à distance notre base de données, ceci est intéressant car on a un aperçu visuel direct avec notre base de données. De plus, on peut visualiser directement nos tables et leur contenu.

Visual Studio Code : Visual Studio Code est un IDE sur lequel nous pouvons installer diverses extensions, on peut personnaliser comme bon nous semble l'editeur pour avoir un environnement confortable pour coder.

² Image : template de conteneur en lecture uniquement contenant un système de base et une application

Docker automatisé de la création de notre environnement

Afin de faciliter l'installation de notre environnement nous avons décidé d'utiliser un fichier docker-compose qui est un fichier en .yaml et qui permet de lancer différents conteneurs.

```
version: '3.8'

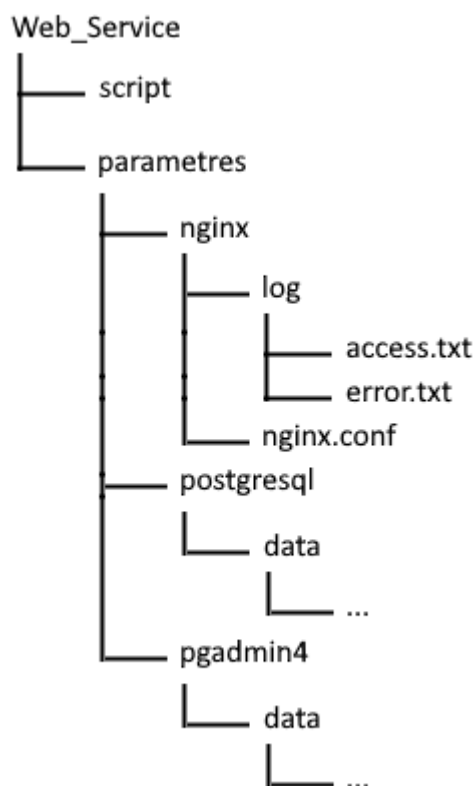
services:
  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin4_SAE
    volumes:
      - "./Web_Service/parametres/pgadmin4/data:/var/lib/pgadmin"
    ports:
      - "8890:443"
      - "8891:80"
    environment:
      PGADMIN_DEFAULT_EMAIL: fatih.fidan@edu.univ-paris13.fr
      PGADMIN_DEFAULT_PASSWORD: MonMDP123soleil&*

  nginx:
    image: nginx:latest
    container_name: nginx_SAE
    volumes:
      - "./Web_Service/script:/usr/share/nginx/html:ro"
      - "./Web_Service/parametres/nginx/log:/var/log/nginx"
      - "./Web_Service/parametres/nginx/nginx.conf:/etc/nginx/nginx.conf:ro"
    ports:
      - "8892:80"

  php:
    image: php:fpm-alpine
    container_name: php_SAE
    volumes:
      - "./Web_Service/script:/script:ro"
    ports:
      - "8893:9000"

  postgresql:
    image: postgres:latest
    container_name: postgresql_SAE
    volumes:
      - "./Web_Service/parametres/postgresql/data:/var/lib/postgresql/data"
    ports:
      - "8894:5432"
    environment:
      POSTGRES_USER: IUT_Villetaneuse
      POSTGRES_PASSWORD: sae_postgres_IUT_villetaneuse;
      POSTGRES_DB: SAE
```

Le fichier docker-compose va nous créer différents répertoire grâce à la ligne “volumes³” du fichier .yaml, la particularité des volumes est qu'ils seront liés entre notre machine physique et le service lancé. Voici le schéma des répertoires qui seront créés automatiquement.



Avec le fichier yaml on definit egalement des ports sur lesquelles nous pourrons accéder a nos conteneurs, nous pouvons également mettre en paramètre des environnements qui définirons différents réglage sur nos conteneurs par exemple sur postgres nous avons créé un utilisateur IUT_Villetaneuse et qui a une base de donnée SAE.

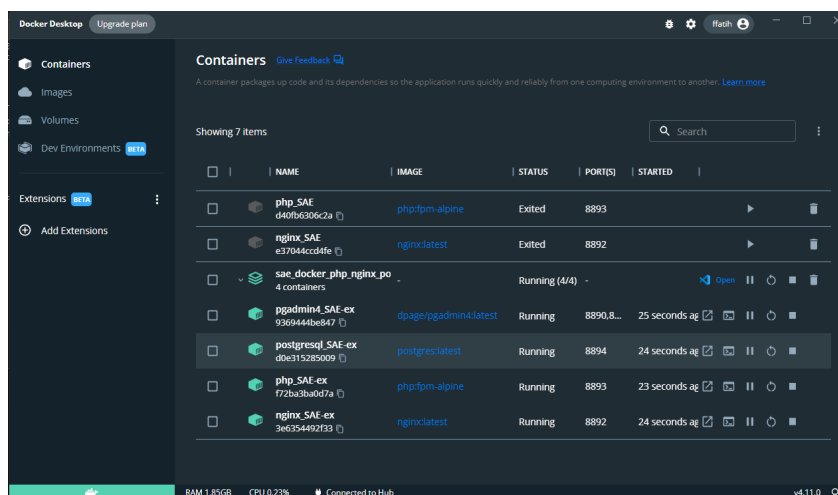
Enfin afin de créer tout ca automatiquement nous executons la commande :

```
docker-compose up -d
```

```

Microsoft Windows [version 10.0.22000.918]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Fatih>cd Desktop
C:\Users\Fatih\Desktop>cd SAE_Docker_PHP_NGINX_POSTGRES_PGADMIN
C:\Users\Fatih\Desktop\SAE_Docker_PHP_NGINX_POSTGRES_PGADMIN>docker-compose up -d
Creating network "sae_docker_php_nginx_postgres_pgadmin_default" with the default driver
Creating pgadmin4_SAE-ex ... done
Creating php_SAE-ex ... done
Creating nginx_SAE-ex ... done
Creating postgresql_SAE-ex ... done
C:\Users\Fatih\Desktop\SAE_Docker_PHP_NGINX_POSTGRES_PGADMIN>
  
```

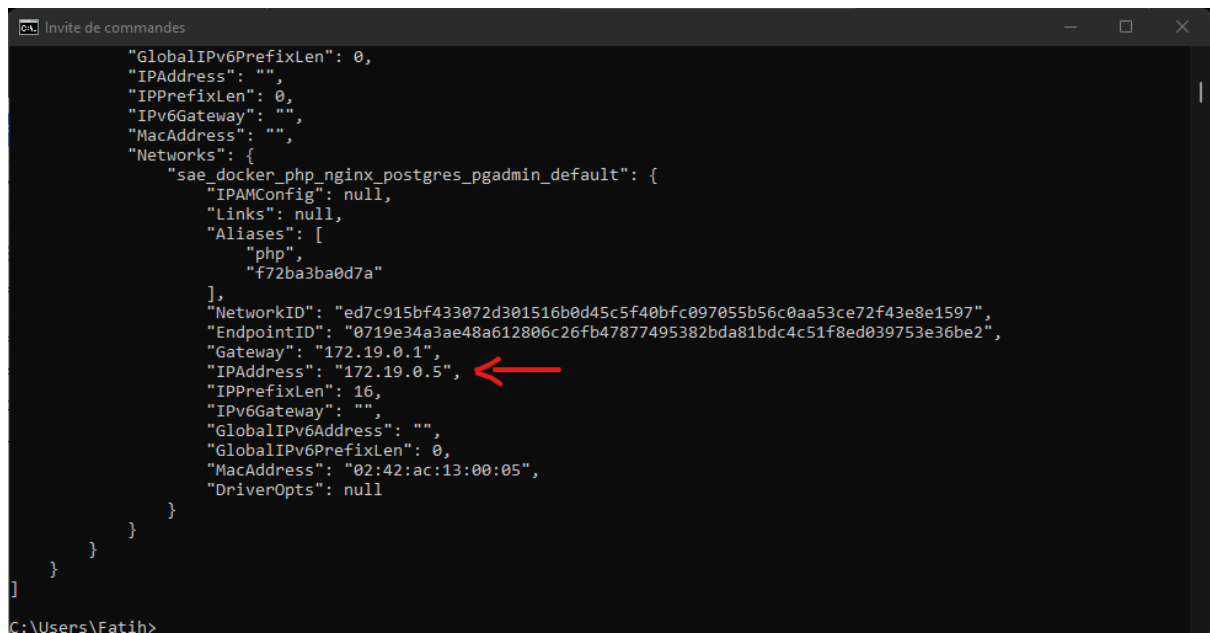


³ Volumes : point de montage qui permet de persister les données.

Lien entre nginx et php

Pour que nginx puisse lire les pages web en .php, il faut le configurer en y ajoutant l'adresse ip du conteneur de docker dans le fichier "nginx.conf" qui se trouve dans "/Web_Service/parametres/nginx". Puis pour obtenir l'ip du conteneur nous utilisons la commande :

```
docker inspect <nom du conteneur>
```



```
Invite de commandes

"GlobalIPv6PrefixLen": 0,
"IPAddress": "",
"IPPrefixLen": 0,
"IPv6Gateway": "",
"MacAddress": "",
"Networks": {
  "sae_docker_php_nginx_postgres_pgadmin_default": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": [
      "php",
      "f72ba3ba0d7a"
    ],
    "NetworkID": "ed7c915bf433072d301516b0d45c5f40bfc097055b56c0aa53ce72f43e8e1597",
    "EndpointID": "0719e34a3ae48a612806c26fb47877495382bda81bdc4c51f8ed039753e36be2",
    "Gateway": "172.19.0.1",
    "IPAddress": "172.19.0.5",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:ac:13:00:05",
    "DriverOpts": null
  }
}
```

C:\Users\Fatih>

Notre fichier nginx.conf ressemble à ceci :

```
user                nginx;
worker_processes    1;

error_log           /var/log/nginx/error.log warn;
pid                /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include          /etc/nginx/mime.types;
    default_type     application/octet-stream;

    log_format        main '$remote_addr - $remote_user [$time_local]
"$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
"$http_x_forwarded_for"';
    access_log        /var/log/nginx/access.log main;

    sendfile          on;
    keepalive_timeout 65;
    server_tokens     off;

    server {
        listen        80;
        server_name    localhost;

        location / {
            root        /usr/share/nginx/html;
            index        index.php index.html index.htm;
        }

        error_page     500 502 503 504 /50x.html;
        location = /50x.html {
            root        /usr/share/nginx/html;
        }

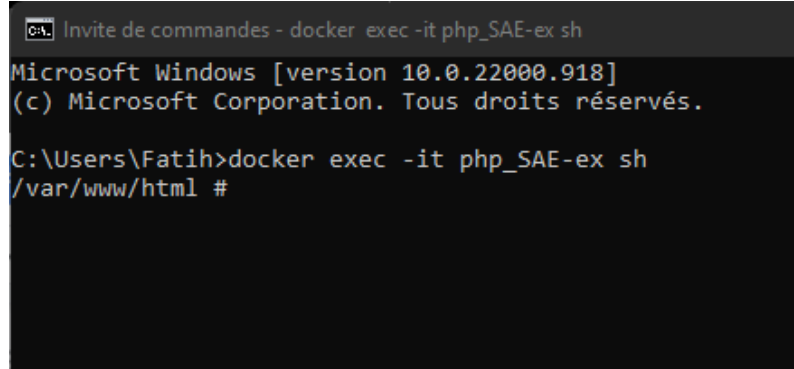
        location ~ /\.php$ {
            root        /usr/share/nginx/html;
            include      fastcgi_params;
            fastcgi_pass 172.19.0.5:9000;
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME /script$fastcgi_script_name;
        }
    }
}
```

Et voilà maintenant notre conteneur nginx peut lire des sites web en .php

Lien entre php et postgresql

Pour pouvoir nous connecter à notre base de données depuis un site web il faut relier php et postgres pour cela nous devons installer des extensions sur php. Pour installer les extensions nous allons ouvrir le terminal de notre conteneur pour ceci nous exécutons la commande :

```
docker exec -it php_SAE-ex sh
```



```
Invite de commandes - docker exec -it php_SAE-ex sh
Microsoft Windows [version 10.0.22000.918]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Fatih>docker exec -it php_SAE-ex sh
/var/www/html #
```

Voici la liste des commandes à exécuter afin d'installer les extensions nécessaire :

```
apk update && apk upgrade
apk add --no-cache postgresql-dev
docker-php-ext-install -j$(nproc) pgsql
docker-php-ext-install -j$(nproc) pdo_pgsql
```

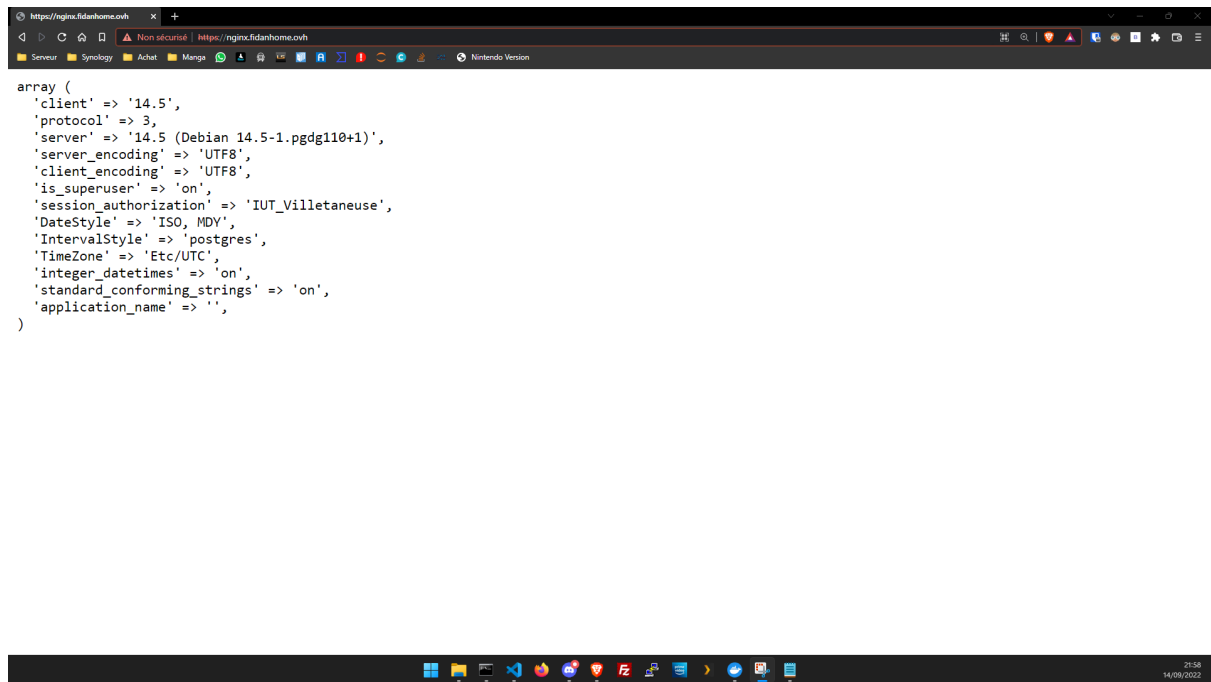
Nous allons maintenant créer un fichier index.php qui va nous permettre de nous connecter à notre base de données.

```
<?php
$dbconn = pg_connect('host=192.168.1.100 port=8894 dbname=SAE
user=IUT_Villetaneuse password=sae_postgres_IUT_villetaneuse;')
    or die('Could not connect');

echo '<pre>' . var_export(pg_version($dbconn), true) . '</pre>';

pg_close($dbconn);
?>
```

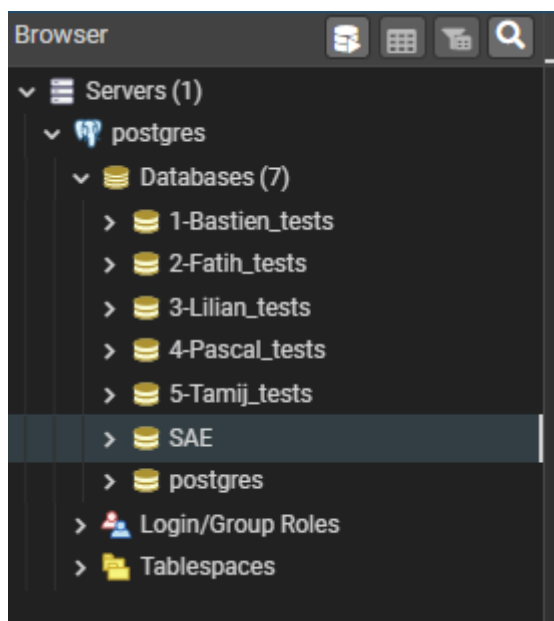
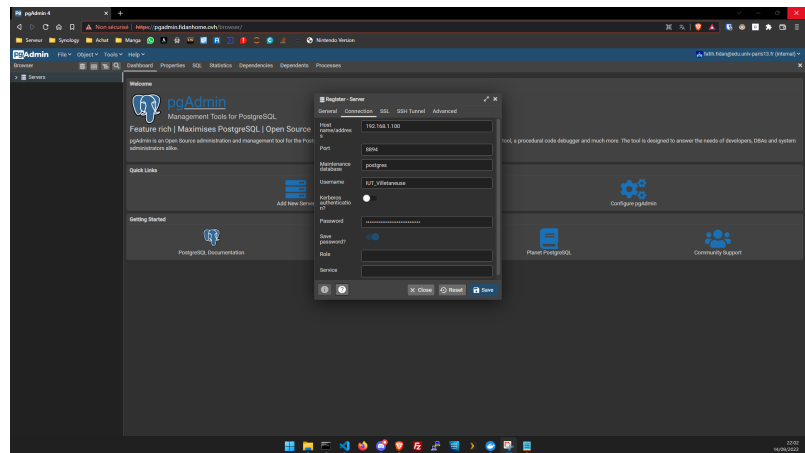
Voici le résultat de la page, nous avons des informations sur notre base de données et nous sommes bien connectés à la base de données.



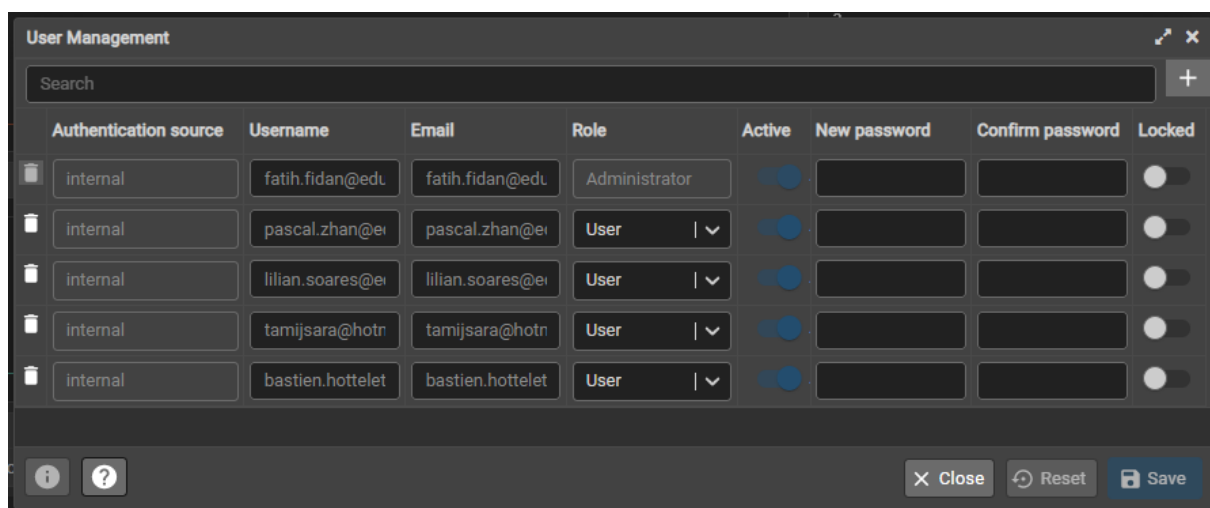
Associer pgadmin4 et postgresql

Avec pgadmin nous nous connectons à notre base de données postgres comme ci-dessous.

Nous avons chacun une base de données de test et une base de données SAE qui regroupe la version finale de la base de données pour notre application web.



De plus, nous avons chacun un accès différent à pgadmin et nous pouvons travailler sur le même environnement à distance.&



*