

NFP 121 - Programmation avancée

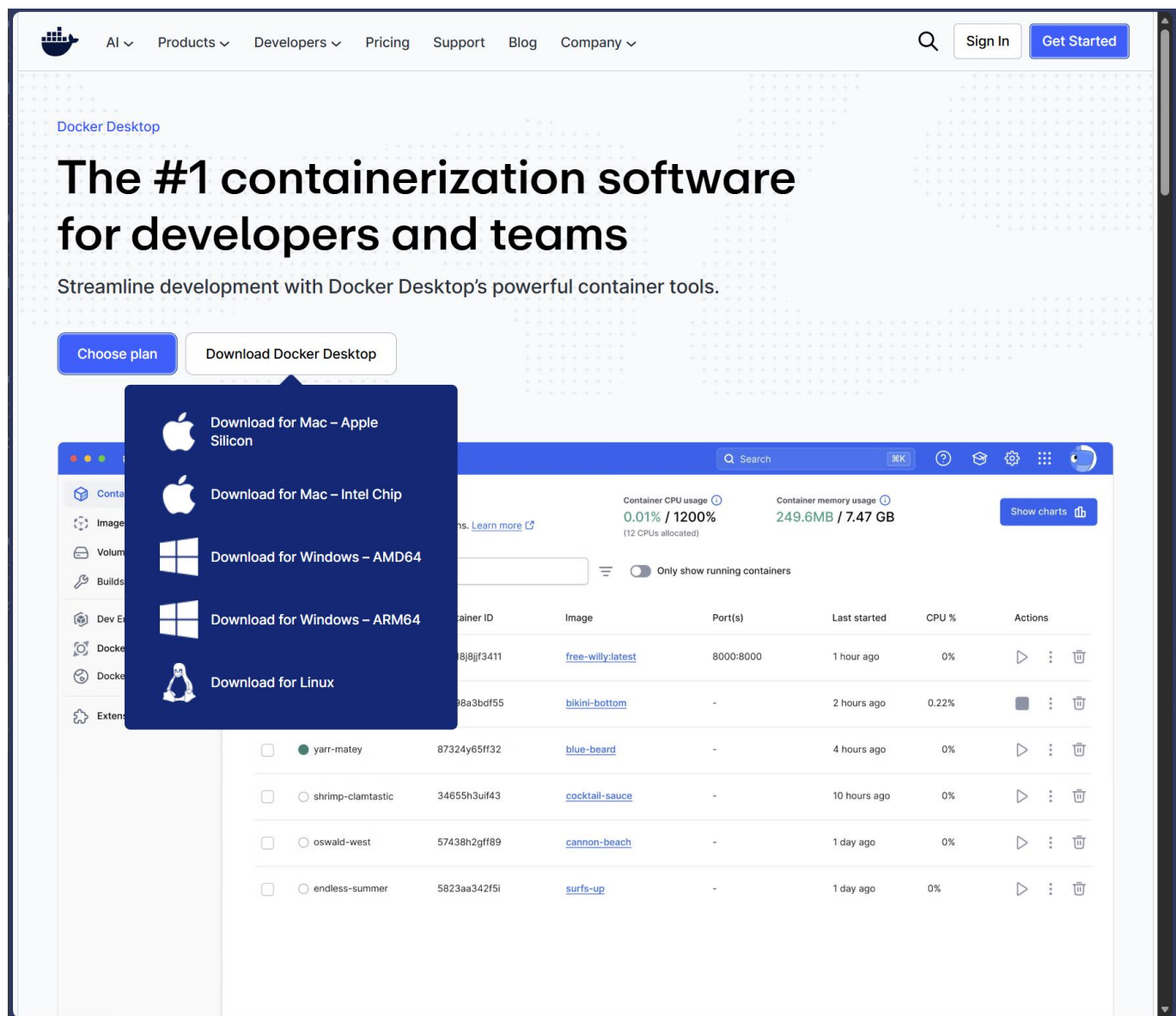
Projet

Installation et configuration des outils

Docker desktop pour Windows

Télécharger et installer Docker desktop pour Windows via l'URL :

[Docker Desktop: The #1 Containerization Tool for Developers | Docker](#)



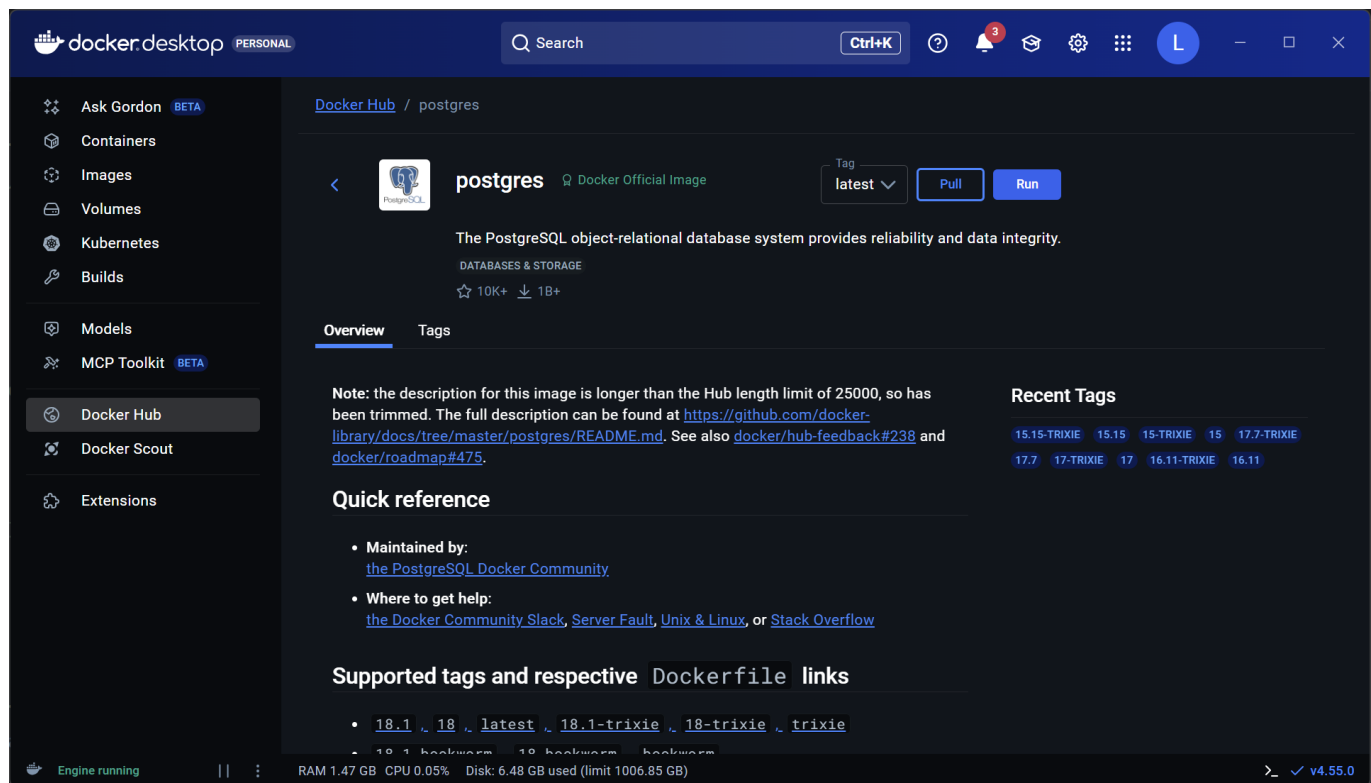
The screenshot shows the Docker Desktop website. The main heading is "The #1 containerization software for developers and teams". Below it, a subheading reads "Streamline development with Docker Desktop's powerful container tools." There are two buttons: "Choose plan" and "Download Docker Desktop". The "Download Docker Desktop" button is highlighted, and a dark blue dropdown menu is open, showing five options with corresponding icons: "Download for Mac – Apple Silicon", "Download for Mac – Intel Chip", "Download for Windows – AMD64", "Download for Windows – ARM64", and "Download for Linux". The background of the website shows a Docker Desktop interface with a sidebar on the left containing "Containers", "Images", "Volumes", "Builds", "Dev Environments", "Docker Desktop", and "Extensions". The main area displays container statistics: "Container CPU usage 0.01% / 1200% (12 CPUs allocated)" and "Container memory usage 249.6MB / 7.47 GB". Below this is a table of running containers.

Container ID	Image	Port(s)	Last started	CPU %	Actions
8j8jif3411	free-willy:latest	8000-8000	1 hour ago	0%	▶ ⋮ 🗑
98a3bdf55	bikini-bottom	-	2 hours ago	0.22%	▶ ⋮ 🗑
87324y65ff32	blue-beard	-	4 hours ago	0%	▶ ⋮ 🗑
34655h3uif43	cocktail-sauce	-	10 hours ago	0%	▶ ⋮ 🗑
57438h2gff89	cannon-beach	-	1 day ago	0%	▶ ⋮ 🗑
5823aa342f5i	surfs-up	-	1 day ago	0%	▶ ⋮ 🗑

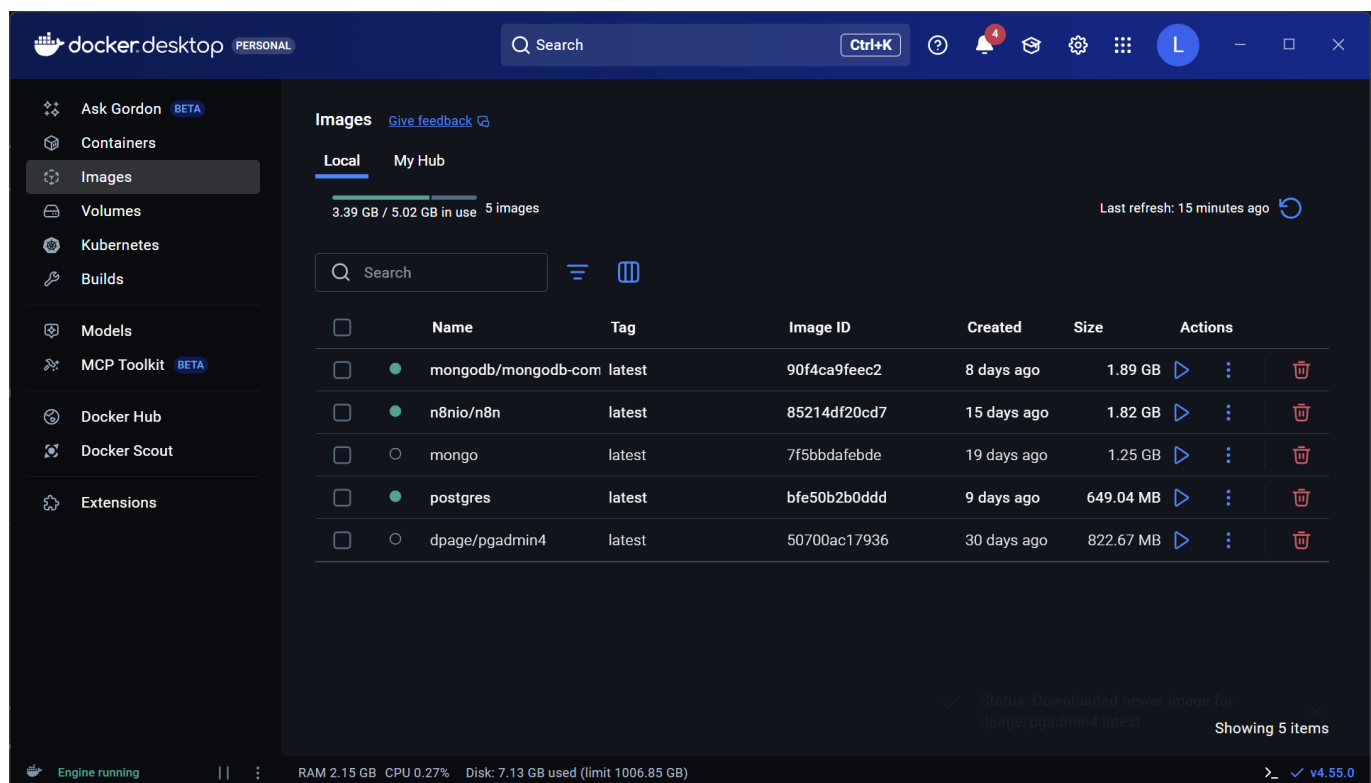
PostgreSQL sous docker

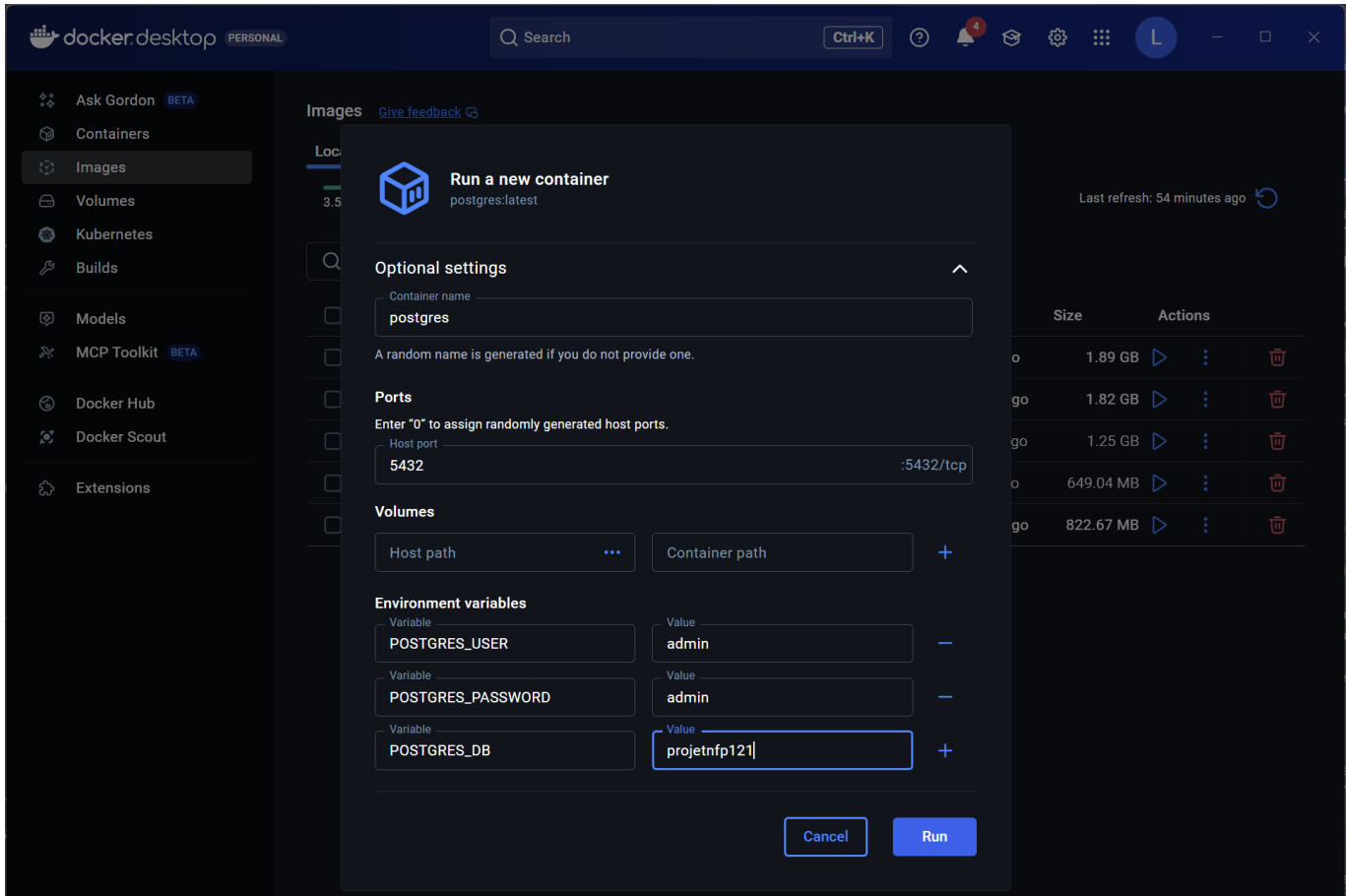
Dans Docker desktop, télécharger (Pull) l'image docker de **postgres** via l'url :

[postgres - Official Image | Docker Hub](#)



Exécuter l'image docker de **postgres** (crée et exécute un container)

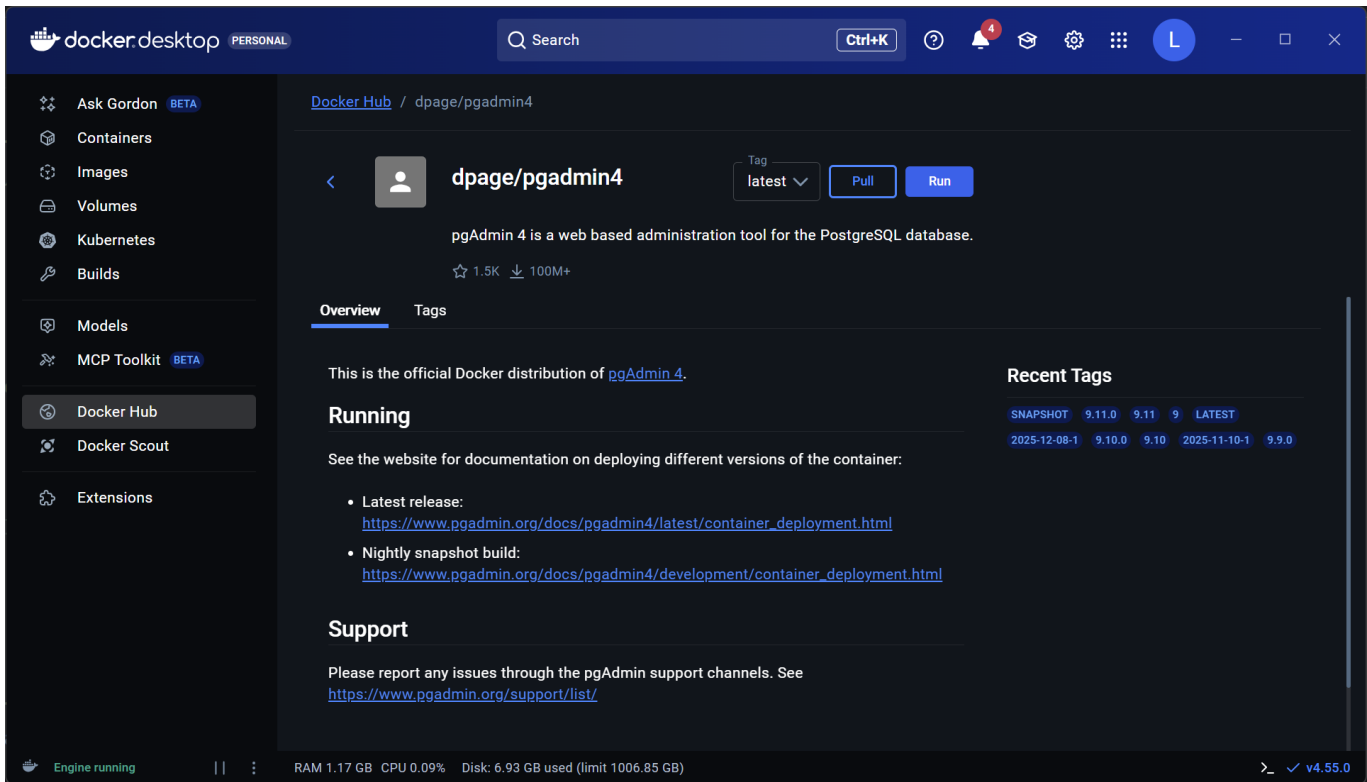




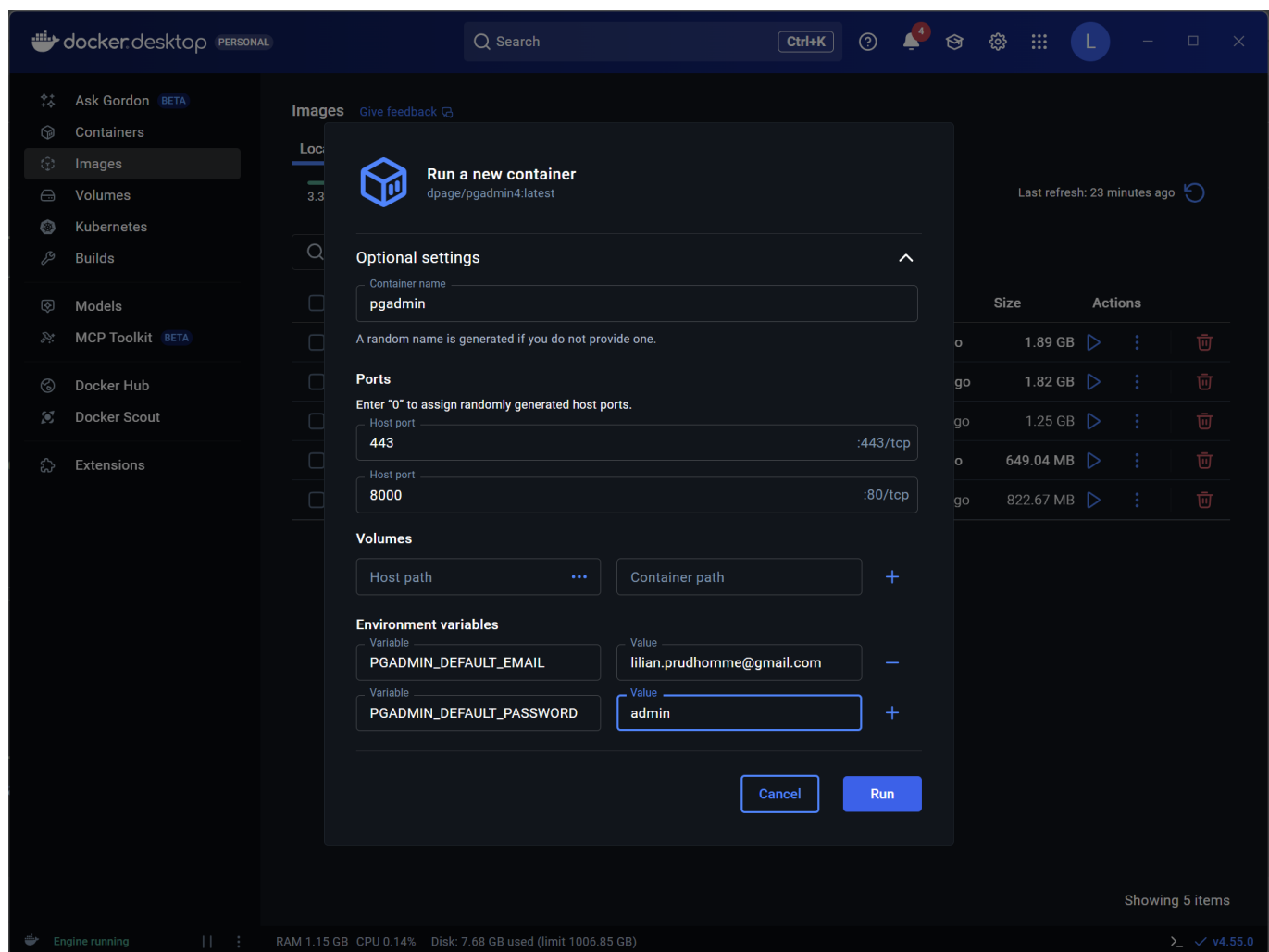
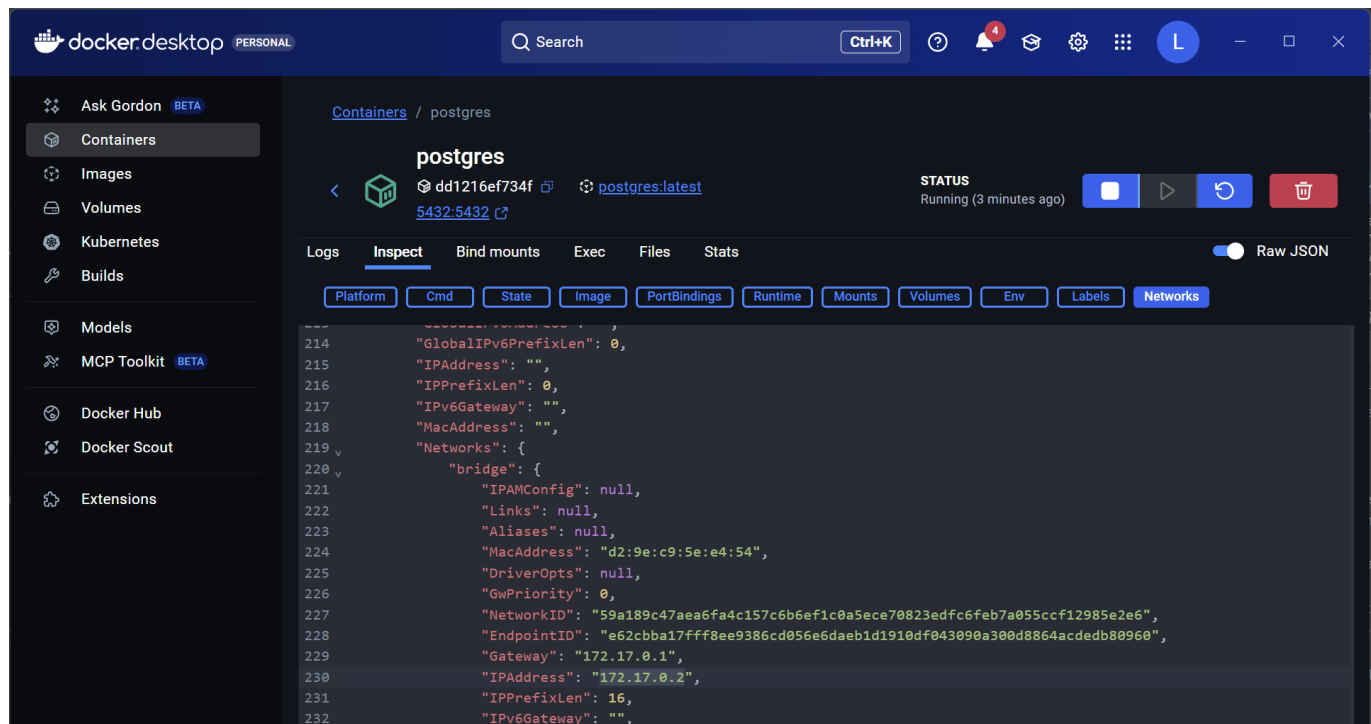
PGAdmin4 sous docker

Dans Docker desktop, télécharger (Pull) l'image docker de **pgadmin4** via l'url :

[dpage/pgadmin4 - Docker Image](https://hub.docker.com/r/dpage/pgadmin4)

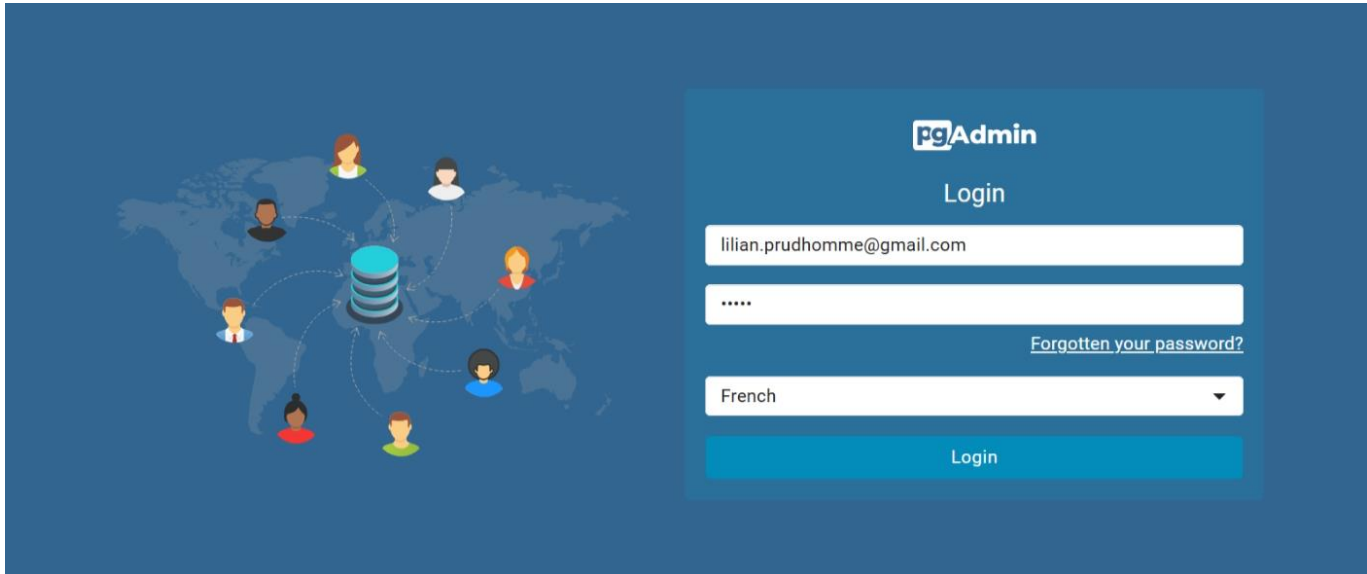


Exécuter l'image docker de **pgadmin4** (crée et exécute un container)

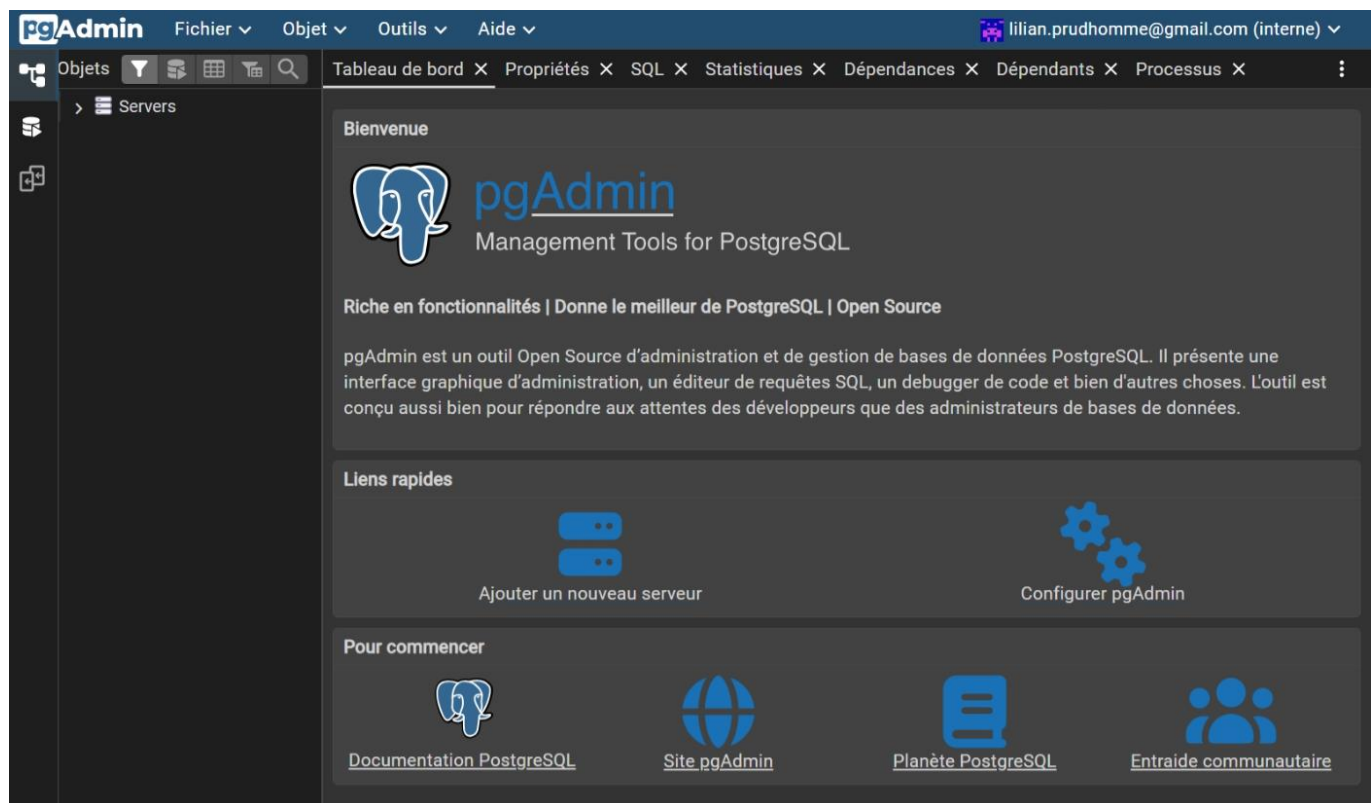


Configurer **pgadmin4** via l'url :

<http://localhost:8000/>



Ajouter le serveur **postgres** de docker :



Nouveau - Serveur

Général

Connexion

Paramètres

Tunnel SSH

Avancé

Post Connection SQL

Tags

Nom

local

Groupe de serveurs

Servers

Arrière plan

X

Premier plan

X

Connecter maintenant ?

Partagé ?

Identifiant Partagé

Commentaires

Le Nom d'hôte ou le Service doit être précisé.

Fermer

Réinitialiser

Enregistrer

Nouveau - Serveur

Général

Connexion

Paramètres

Tunnel SSH

Avancé

Post Connection SQL

Tags

Nom d'hôte / Adresse

172.17.0.2

Port

5432

Base de données de maintenance

projetnfp121

Identifiant

admin

Authentification Kerberos ?

Mot de passe

.....

In edit mode the password field is enabled only if Save Password is set to true.

Enregistrer le mot de passe ?

Rôle

Service

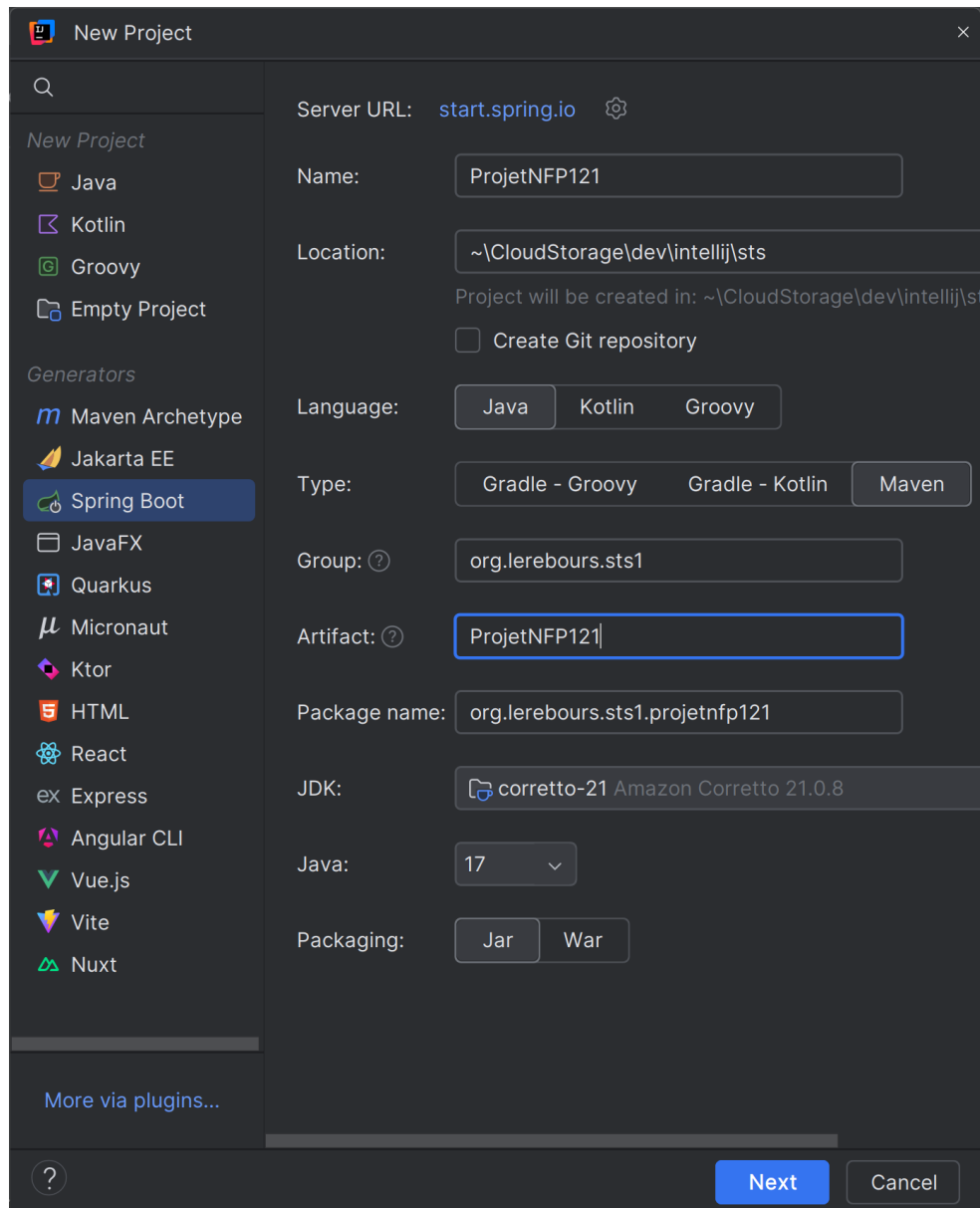
Fermer

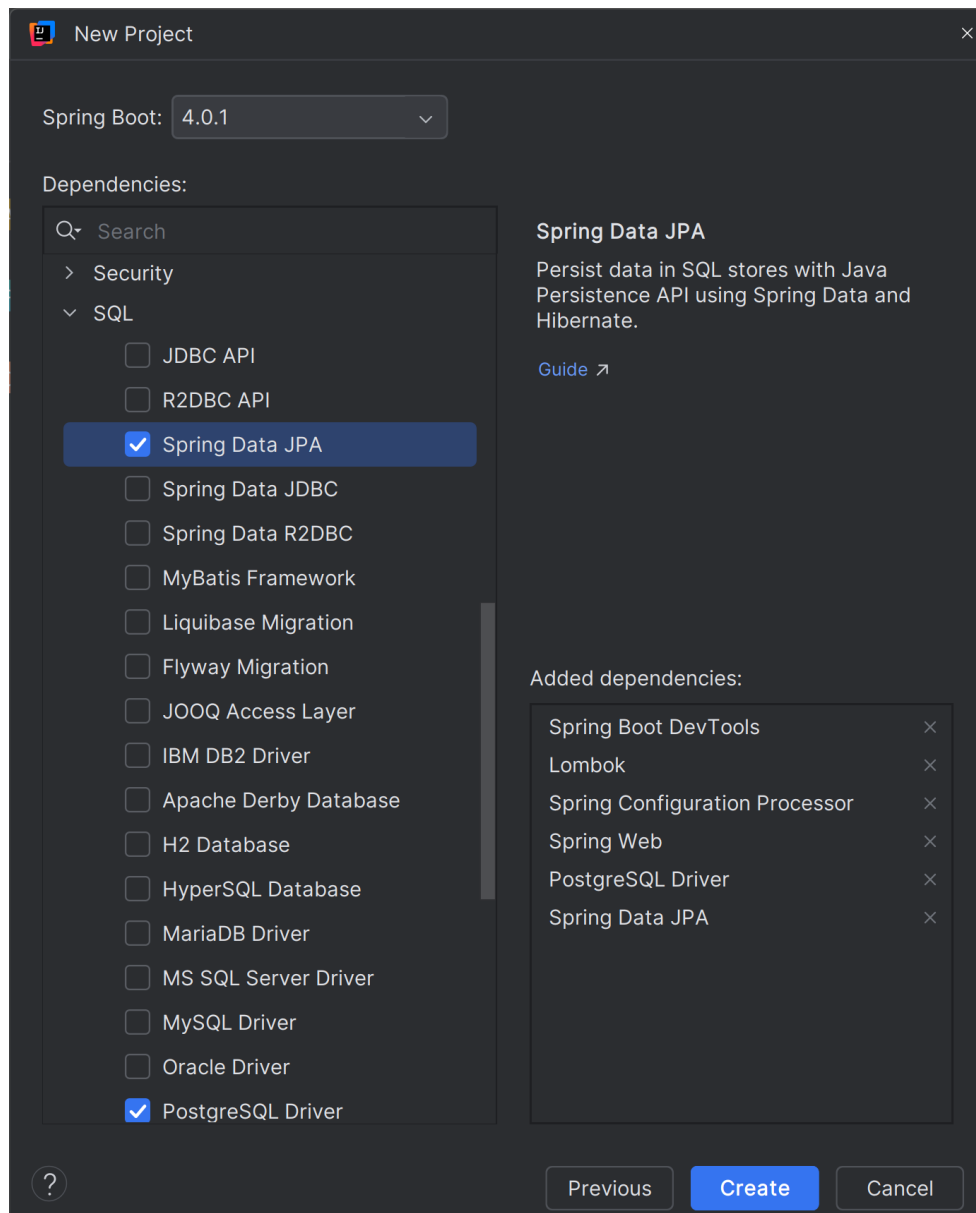
Réinitialiser

Enregistrer

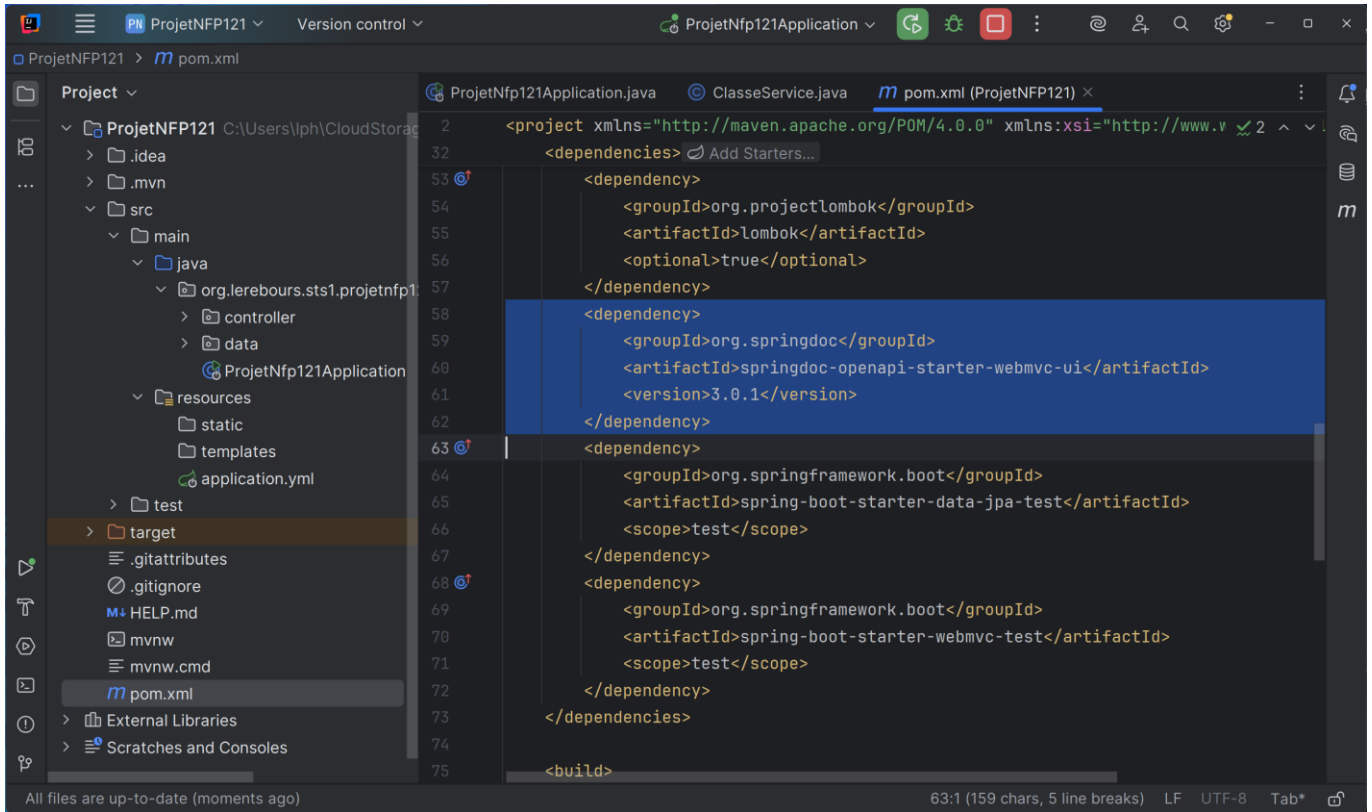
Création et configuration du projet Java SpringBoot (avec swagger)

Dans IntelliJ IDEA, créer un nouveau projet **Spring Boot**





Dans le fichier pom.xml (maven), ajouter la dépendance pour swagger :



Implémenter le projet

Configuration initiale

Modifier le fichier **src/main/java/org/lerebours/sts1/projetnfp121/ProjetNfp121Application.java**

```
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProjetNfp121Application {

    public static void main(String[] args) {
        System.out.println("Démarrage du Back-end !!!!!!!!!!!");
        SpringApplication.run(ProjetNfp121Application.class, args);
    }
}
```

Créer le fichier **src/main/resources/application.yml** (remplace le fichier **application.properties**)

```
spring:
  application:
    name: ProjetNFP121

  datasource:
    url: jdbc:postgresql://127.0.1.2:5432/projetnfp121
    username: admin
    password: admin
    driver-class-name: org.postgresql.Driver

  jpa:
    hibernate:
      ddl-auto: validate # dev: update / create-drop, prod: validate/none
    properties:
      hibernate:
        format_sql: true
        open-in-view: false

logging:
  level:
    org.hibernate.SQL: debug
    org.hibernate.orm.jdbc.bind: trace # Boot 3 / Hibernate 6 (paramètres SQL)
```

Implémentation des @Entity (objet Java ↔ DB)

Créer le package **data** et créer dedans la classe **Classe**

```
package org.lerebours.sts1.projetnfp121.data;

import jakarta.persistence.*;
import lombok.Data;

@Data
@Entity
public class Classe {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String denomination;
}
```

Dans le package **data**, créer la classe **Etudiant**

```
package org.lerebours.sts1.projetnfp121.data;

import jakarta.persistence.*;
import lombok.Data;

@Data
@Entity
public class Etudiant {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String nom;

    @Column(nullable = false)
    private String prenom;

    // private String photo;

    @ManyToOne(fetch = FetchType.EAGER
) // ne charge la classe que si elle est utilisée
    // @JoinColumn(name = "classe_id") // facultatif si le champ de la clé étrangère
    se nomme <champ>_id
    private Classe classe;
}
```

Implémentation des Repository (gestion des @Entity)

Dans le package **data**, créer l'interface **ClasseRepository**

```
package org.lerebours.sts1.projetnfp121.data;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ClasseRepository extends JpaRepository<Classe, Long> {
}
```

Dans le package **data**, créer l'interface **EtudiantRepository**

```
package org.lerebours.sts1.projetnfp121.data;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {
    List<Etudiant> getEtudiantsByClasse_Id(Long classeId);
}
```

Implémentation des @RestController (API Rest)

Créer le package **controller** et créer dedans la classe **ClasseService**

```
package org.lerebours.sts1.projetnfp121.controller;

import org.lerebours.sts1.projetnfp121.data.Classe;
import org.lerebours.sts1.projetnfp121.data.ClasseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/classes")
class ClasseService {

    @Autowired private ClasseRepository classeRepository;

    @GetMapping("/")
    List<Classe> getAllClasses() {
        return classeRepository.findAll();
    }

    @GetMapping("/{id}")
    Classe getClasseById(@PathVariable Long id) {
        return classeRepository.findById(id).orElse(null);
    }

    @PostMapping("/add" )
    void addClasse(@RequestBody Classe classe) {
        classeRepository.save(classe);
    }

}
```

Dans le package **controller**, créer la classe **EtudiantService**

```
package org.lerebours.sts1.projetnfp121.controller;

import org.lerebours.sts1.projetnfp121.data.Etudiant;
import org.lerebours.sts1.projetnfp121.data.EtudiantRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/etudiants")
class EtudiantService {

    @Autowired private EtudiantRepository etudiantRepository;

    @GetMapping("/")
    List<Etudiant> getAllEtudiants() {
        return etudiantRepository.findAll();
    }

    @GetMapping("/{id}")
    Etudiant getEtudiantById(@PathVariable Long id) {
        return etudiantRepository.findById(id).orElse(null);
    }

    @GetMapping("/classe/{classe_id}")
    List<Etudiant> getEtudiantsByClasse(@PathVariable Long classe_id) {
        return etudiantRepository.getEtudiantsByClasse_Id(classe_id);
    }

}
```

```
}

@GetMapping("/disponible")
List<Etudiant> getEtudiantsDisponibles() {
    return getEtudiantsByClasse(null);
}

@PostMapping("/add" )
void addEtudiant(@RequestBody Etudiant etudiant) {
    etudiantRepository.save(etudiant);
}

}
```

Exécuter l'application

Tester les API du Back-end via swagger UI

Dans votre navigateur, ouvrir l'url : <http://localhost:8080/swagger-ui/index.html>

The screenshot displays the Swagger UI interface for an API. At the top, the Swagger logo is visible, along with the text 'Supported by SMARTBEAR'. The URL bar shows '/v3/api-docs' and an 'Explore' button. Below this, the 'OpenAPI definition' is shown for version 'v0' using 'OAS 3.1'. A 'Servers' section indicates the server URL as 'http://localhost:8080 - Generated server url'. The main content area lists two services: 'etudiant-service' and 'classe-service'. Each service has a list of endpoints with their respective HTTP methods (POST, GET) and paths. For 'etudiant-service', the endpoints are: POST /etudiants/add, GET /etudiants/{id}, GET /etudiants/disponible, GET /etudiants/classe/{classe_id}, and GET /etudiants/. For 'classe-service', the endpoints are: POST /classes/add, GET /classes/{id}, and GET /classes/. At the bottom, a 'Schemas' section shows two schemas: 'Classe' and 'Etudiant', both of type 'object'.

Swagger
Supported by SMARTBEAR

/v3/api-docs Explore

OpenAPI definition v0 OAS 3.1
/v3/api-docs

Servers
http://localhost:8080 - Generated server url

etudiant-service

- POST /etudiants/add
- GET /etudiants/{id}
- GET /etudiants/disponible
- GET /etudiants/classe/{classe_id}
- GET /etudiants/

classe-service

- POST /classes/add
- GET /classes/{id}
- GET /classes/

Schemas

- Classe > Expand all object
- Etudiant > Expand all object

Test de <http://localhost:8080/classes/>

GET /classes/

Parameters

Cancel

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/classes/' \
-H 'accept: */*'
```

Request URL

http://localhost:8080/classes/

Server response

Code	Details
200	<div><div>Response body<pre>[{ "denomination": "STS1 2025-2026", "id": 1 }]</pre></div><div>Download</div><div>Response headers<pre>connection: keep-alive content-type: application/json date: Wed, 07 Jan 2026 18:20:08 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre></div></div>

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "denomination": "string"
  }
]
```

15

Test de <http://localhost:8080/classes/1>

GET

/classes/{id}

Parameters

Cancel

Name	Description
id * required integer(\$int64) (path)	<input type="text" value="1"/>

Execute

Clear

Responses

Curl

curl -X 'GET' \
'http://localhost:8080/classes/1' \
-H 'accept: */*'

Request URL

http://localhost:8080/classes/1

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "denomination": "STS1 2025-2026", "id": 1 }</pre><div> Download</div></div> <div>Response headers</div> <div>connection: keep-alive content-type: application/json date: Wed, 07 Jan 2026 18:23:16 GMT keep-alive: timeout=60 transfer-encoding: chunked</div>

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

```
{  
  "id": 0,  
  "denomination": "string"  
}
```


Test de <http://localhost:8080/classes/add>

classe-service

POST

/classes/add

Parameters

Cancel

Reset

No parameters

Request body required

application/json

Edit Value | Schema

```
{
  "denomination": "STS2 20254-2026"
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/classes/add' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "denomination": "STS2 20254-2026"
  }'
```

Request URL

http://localhost:8080/classes/add

Server response

Code	Details
200	<div><div>Response headers</div><pre>connection: keep-alive content-length: 0 date: Wed, 07 Jan 2026 18:25:58 GMT keep-alive: timeout=60</pre></div>

Responses

Code	Description	Links
200	OK	No links

Test de <http://localhost:8080/etudiants/>

GET

/etudiants/

⌵

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/etudiants/' \
-H 'accept: */*'
```

⌵

Request URL

http://localhost:8080/etudiants/

Server response

Code

Details

200

Response body

```
[
  {
    "classe": null,
    "id": 1,
    "nom": "MARTIN",
    "prenom": "PIERRE"
  },
  {
    "classe": {
      "denomination": "STS1 2025-2026",
      "id": 1
    },
    "id": 2,
    "nom": "DUPONT",
    "prenom": "JEAN"
  }
]
```

⌵

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed, 07 Jan 2026 19:32:01 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code

Description

Links

200

OK

No links

Media type

/

⌵

Controls Accept header.

Example Value

Schema

```
[
  {
    "id": 0,
    "nom": "string",
    "prenom": "string",
    "classe": {
      "id": 0,
      "denomination": "string"
    }
  }
]
```

Test de <http://localhost:8080/etudiants/classe/1>

GET

/etudiants/classe/{classe_id}

Parameters

Cancel

Name

Description

classe_id * required

integer(\$int64)

(path)

1

Execute

Clear

Responses

Curl

curl -X 'GET' \

'http://localhost:8080/etudiants/classe/1' \

-H 'accept: */*'

Request URL

http://localhost:8080/etudiants/classe/1

Server response

Code

Details

200

Response body

[

{

"classe": {

"denomination": "STS1 2025-2026",

"id": 1

}

,

"id": 2,

"nom": "DUPONT",

"prenom": "JEAN"

}

]

Response headers

connection: keep-alive

content-type: application/json

date: Wed, 07 Jan 2026 19:37:20 GMT

keep-alive: timeout=60

transfer-encoding: chunked

Responses

Code

Description

Links

200

OK

No links

Media type

/

Controls Accept header.

Example Value

Schema

[

{

"id": 0,

"nom": "string",

"prenom": "string",

"classe": {

"id": 0,

"denomination": "string"

}

}

]

A vous de « jouer » ...

19