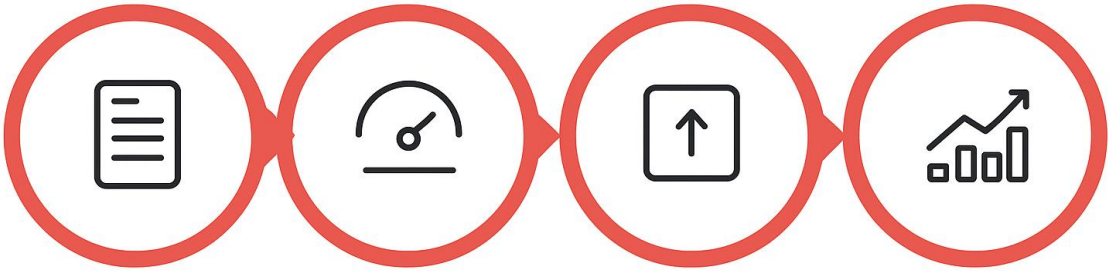
MISE A DISPOSITION DES SERVICES

BTS SIO 2025-2026

SOMMAIRE



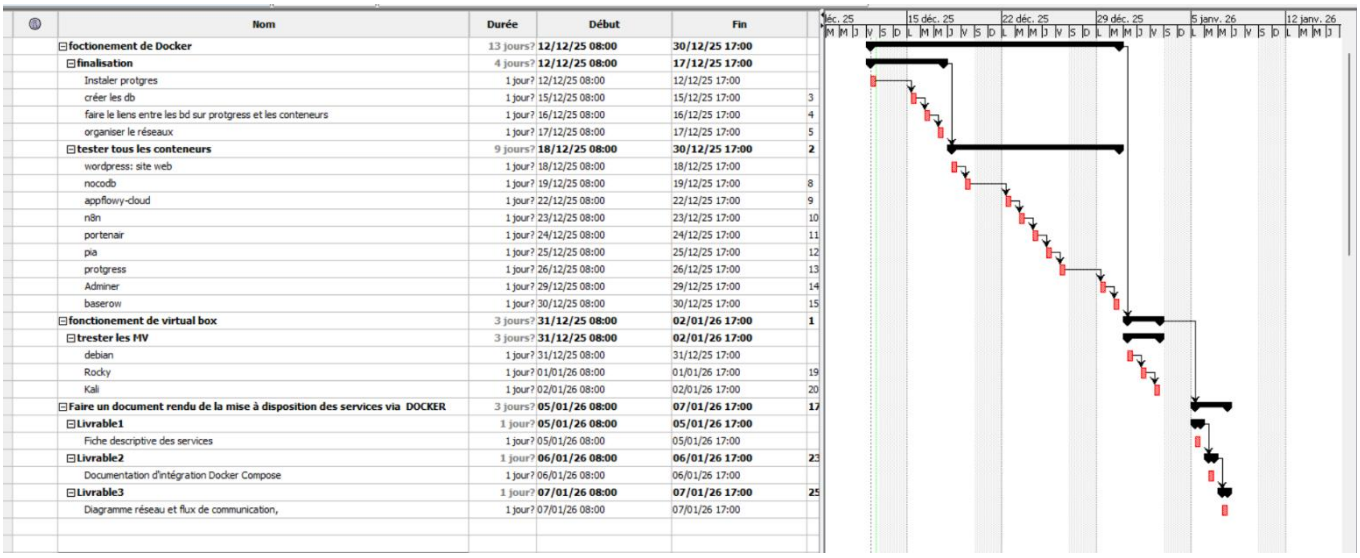
CONTEXTE

Ce document formalise les travaux de déploiement d'applications via Docker et Docker Compose pour le module "Support réseau des accès utilisateurs".

Contexte et Objectifs

- Le but est de documenter les déploiements d'applications réalisés depuis le début de l'année scolaire.

Plan de travail



	Nom	Durée	Début	Fin	
⊟	**foctionement de Docker**	13 jours?	**12/12/25 08:00**	**30/12/25 17:00**	
⊟	**finalisation**	4 jours?	**12/12/25 08:00**	**17/12/25 17:00**	
	Instaler protgres	1 jour?	12/12/25 08:00	12/12/25 17:00	
	créer les db	1 jour?	15/12/25 08:00	15/12/25 17:00	3
	faire le liens entre les bd sur protgress et les conteneurs	1 jour?	16/12/25 08:00	16/12/25 17:00	4
	organiser le réseaux	1 jour?	17/12/25 08:00	17/12/25 17:00	5
⊟	**tester tous les conteneurs**	9 jours?	**18/12/25 08:00**	**30/12/25 17:00**	2
	wordpress: site web	1 jour?	18/12/25 08:00	18/12/25 17:00	
	nocodb	1 jour?	19/12/25 08:00	19/12/25 17:00	8
	appflowy-cloud	1 jour?	22/12/25 08:00	22/12/25 17:00	9
	n8n	1 jour?	23/12/25 08:00	23/12/25 17:00	10
	portenair	1 jour?	24/12/25 08:00	24/12/25 17:00	11
	pia	1 jour?	25/12/25 08:00	25/12/25 17:00	12
	protgress	1 jour?	26/12/25 08:00	26/12/25 17:00	13
	Adminer	1 jour?	29/12/25 08:00	29/12/25 17:00	14
	baserow	1 jour?	30/12/25 08:00	30/12/25 17:00	15
⊟	**fonctionement de virtual box**	3 jours?	**31/12/25 08:00**	**02/01/26 17:00**	1
⊟	**trester les MV**	3 jours?	**31/12/25 08:00**	**02/01/26 17:00**	
	debian	1 jour?	31/12/25 08:00	31/12/25 17:00	
	Rocky	1 jour?	01/01/26 08:00	01/01/26 17:00	19
	Kali	1 jour?	02/01/26 08:00	02/01/26 17:00	20
⊟	**Faire un document rendu de la mise à disposition des services via DOCKER**	3 jours?	**05/01/26 08:00**	**07/01/26 17:00**	17
⊟	**Livrable1**	1 jour?	**05/01/26 08:00**	**05/01/26 17:00**	
	Fiche descriptive des services	1 jour?	05/01/26 08:00	05/01/26 17:00	
⊟	**Livrable2**	1 jour?	**06/01/26 08:00**	**06/01/26 17:00**	23
	Documentation d'intégration Docker Compose	1 jour?	06/01/26 08:00	06/01/26 17:00	
⊟	**Livrable3**	1 jour?	**07/01/26 08:00**	**07/01/26 17:00**	25
	Diagramme réseau et flux de communication,	1 jour?	07/01/26 08:00	07/01/26 17:00	

LIVRABLE1

Ce tableau représente la fiche descriptive des services déployés via Docker

Nom du service	Version	Objectif et fonctionnalité
protgres	15	Serveur de base de données pour stocker et gérer les données des autres applications
baserow	1.21	Plateforme no-code permettant de créer et gérer des bases de données collaboratives
appflowy	Cloud Beta	Outil de prise de notes et gestion de tâches synchronisé dans le cloud
nocodb	0.200	Convertit une base SQL en interface type Airtable pour gérer données et automatisations.
n8n	1.62	Automatisation de workflows permettant de connecter et orchestrer différents services.
wordpress	6.4	destinée à créer et gérer un site web ou blog.
mysql_docker	8.0	Base de données MySQL utilisée par les instances WordPress
portenair	2.20	Interface Portainer permettant d'administrer facilement les conteneurs Docker.
pia	3.7.0	Fournit un service VPN permettant de sécuriser et anonymiser le trafic réseau des autres conteneurs
projector	1.4	Environnement JetBrains accessible via navigateur pour développer à distance

Interdépendance entre les contenaires

- Baserow, n8n et nocodb utilise protgres comme base de données principal
- Workpress utilise MySQL comme base de données principale

LIVRABLE2

Les éléments essentiels de Docker et Docker Compose

a) Pour Docker

- **Image** : modèle de conteneur.
- **Conteneur** : instance d'une image.
- **Dockerfile** : fichier pour créer une image personnalisée.

- **Volume** : stockage persistant pour les données.
- **Réseau Docker** : permet la communication entre conteneurs.

b) Pour Docker Compose

- **services** : conteneurs à lancer.
- **image** ou **build** : image Docker à utiliser ou construire.
- **ports** : ports à exposer sur la machine hôte.
- **volumes** : dossiers partagés entre hôte et conteneur.
- **environment** : variables d'environnement.
- **depends_on** : définit l'ordre de démarrage des services.

Pour déployer nos différents conteneurs nous avons donc des fichier.yml ou fichier.yaml

C'est un fichier de configuration pour Docker Compose, qui permet de définir plusieurs conteneurs et leurs interactions en un seul endroit.
Il ne contient pas le code ou le système d'exploitation : il dit simplement à Docker quelles images utiliser, quels ports ouvrir, quels volumes monter, quels réseaux créer, quelles variables d'environnement appliquer.


```yaml
version: "3"
services:
  wordpress:
    image: wordpress:latest
    restart: always
    ports:
      - "82:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: MySQLUsername
      WORDPRESS_DB_PASSWORD: MySQLUserPassword
      WORDPRESS_DB_NAME: MySQLDatabaseName
    volumes:
      - "./:/var/www/html"
    networks:
      - frontend
networks:
  frontend:
    name: wordpress # Use a custom name
    driver_opts:    # pass options to driver for network creation
      com.docker.network.bridge.host_binding_ipv4: 127.0.0.1
```

On a utilisé la **version 3**

L'image utilisée est **wordpress : latest**

Il utilise le port **82** pour publier ces services et utilise le port **80** en interne

Il utilise MySQL comme serveur de base de données

Le nom de la base de données est : **MySQLDatabaseName**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: wordpress**

Wordpress 1

```yaml
version: "3"
services:
  wordpress:
    image: wordpress:latest
    restart: always
    ports:
      - "81:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: MySQLUsername
      WORDPRESS_DB_PASSWORD: MySQLUserPassword
      WORDPRESS_DB_NAME: wdp2
    volumes:
      - "./:/var/www/html"
    networks:
      - frontend1
networks:
  frontend1:
    name: wordpress1 # Use a custom name
    driver_opts:    # pass options to driver for network creation
      com.docker.network.bridge.host_binding_ipv4: 127.0.0.1
```

On a utilisé la **version 3**

L'image utilisée **est wordpress : latest**

Il utilise le port **81**pour publier ces services et utilise le port **80** en interne

Il utilise **MySQL** comme serveur de base de données

Le nom de la base de données est : **wdp2**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: wordpress1**

MySQL +PHP

```yaml
version: "3"
services:
  db:
    image: mysql:latest
    restart: always
    ports:
    environment:
      MYSQL_ROOT_PASSWORD: MySQLRootPassword
      MYSQL_DATABASE: MySQLDatabaseName
      MYSQL_USER: MySQLUsername
      MYSQL_PASSWORD: MySQLUserPassword
    networks:
      - frontend
      - frontend1
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
```

On a utilisé la **version 3**

L'image utilisée **est mysql:latest**

Il utilise par défaut le port **TCP 3306**

Le nom de la base de données est : **MySQLUserPassword**

Au niveau de l'environnement ce sont les identifiants de connexion

Il utilise deux réseaux **: wordpress et wordpress1**

L'image utilisée **est phpmyadmin/phpmyadmin**

mysql:latest

Projeqtor

```yaml
services:
  projeqtor:
    image: mraber/projeqtor:12.2.3-1
    restart: always
    environment:
      PJQ_PHP_MAX_INPUT_VARS: 4000
      PJQ_PHP_REQUEST_TERMINATE_TIMEOUT: 0
      PJQ_PHP_MAX_EXECUTION_TIME: 30
      PJQ_PHP_MEMORY_LIMIT: "512M"
      PJQ_PHP_MAX_UPLOAD_SIZE: "512M"
      PJQ_DB_TYPE: "mysql"
      PJQ_DB_HOST: "db:3306"
      PJQ_DB_NAME: "projeqtor"
      PJQ_DB_USER: "wdpuser"
      PJQ_DB_PASSWORD: "wdppassword"
      PJQ_DB_DISPLAY_NAME: "Projeqtor"
      PJQ_DB_COLLATION: "utf8mb4_general_ci"
      PJQ_DIR_ATTACHMENTS: "/var/data/projeqtor/attachments"
      PJQ_DIR_DOCUMENTS: "/var/data/projeqtor/documents"
      PJQ_LOG_FILE: "/var/data/projeqtor/logs/projeqtor_$${date}.log"
      # Levels : debug (4) -> info (2)
      PJQ_LOG_LEVEL: 2
    volumes:
      - prj_attachments:/var/data/projeqtor/attachments
      - prj_documents:/var/data/projeqtor/documents
      - prj_logs:/var/data/projeqtor/logs
    ports:
      - 8096:80
```

```yaml
      - prj_logs:/var/data/projeqtor/logs
    ports:
      - 8096:80
    logging:
      # Do not fill disk with logs!
      driver: "json-file"
      options:
        max-size: "10M"
        max-file: "5"
        compress: "true"
    networks:
      - frontend
volumes:
  prj_db_data: {}
  prj_attachments: {}
  prj_documents: {}
  prj_logs: {}
networks:
  frontend:
    name: wordpress # Use a custom name
    driver_opts:    # pass options to driver for network creation
      com.docker.network.bridge.host_binding_ipv4: 127.0.0.1
```

L'image utilisée **est mraber/projeqtor:12.2.3-1**

Il utilise le port **8096** pour publier ces services et utilise le port **80** en interne

Il utilise **MySQL** comme serveur de base de données

Le nom de la base de données est : **Projeqtor**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: wordpress1**

Portainer

```yaml
version: '3'

services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    privileged: true # /!\ Le conteneur est lancé en mode privilégié /!\
    security_opt:
      - no-new-privileges:true
    volumes:
      - /etc/localtime:/etc/localtime:ro # pour synchroniser l'heure avec
      - /var/run/docker.sock:/var/run/docker.sock:ro # socket associé à Do
      - /dossier_docker/portainer/data:/data # données de portainer
    ports:
      - 9000:9000 # port pour l'interface web
```

On a utilisé la **version 3**

L'image utilisée est **portainer/portainer-ce:latest**

Il utilise le port **9000** pour publier ces services et utilise le port **9000** en interne

Le nom du réseau est défini par défaut par Docker

Postgres

```yaml
services:
  postgresdb:
    container_name: postgresdb
    build: .
    restart: always
    environment:
      POSTGRES_USER: naomie
      POSTGRES_PASSWORD: naomiep
      POSTGRES_DB: mybase

    volumes:
      - postgresdb:/var/lib/postgresql/data
    networks:
      - frontend2
  adminer:
    image: adminer
    ports:
      - 8082:8080
    networks:
      - frontend2
volumes:
  postgresdb: {}
networks:
  frontend2:
    name: nocode # Use a custom name
    driver_opts:    # pass options to driver for network creation
      com.docker.network.bridge.host_binding_ipv4: 127.0.0.1
```

L'image utilisée **est postgresdb**

Il utilise par défaut le port **5432**

Le nom de la base de données est : **mybase**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: nocode**

L'image utilisée **adminer**

Il utilise le port **8082** pour publier ces services et utilise le port **8080** en interne

Le nom du réseau est **: nocode**

Baserow

```yaml
version: "3.8"

services:
  baserow:
    image: baserow/baserow:latest
    container_name: baserow_app
    ports:
      - "8000:80"
    restart: unless-stopped
    environment:
      DATABASE_USER: naomie
      DATABASE_PASSWORD: naomiep
      DATABASE_NAME: Baserow
      DATABASE_HOST: postgresdb
      BASEROW_PUBLIC_URL: http://localhost:8000
    volumes:
      - baserow_data:/baserow/data
    networks:
      - frontend2

volumes:
  db_data:
  baserow_data:
networks:
  frontend2:
    name: nocode # Use a custom name
    external: true
```

On a utilisé la **version 3.8**

L'image utilisée est **baserow/baserow:latest**

Il utilise le port **8000** pour publier ces services et utilise le port **80** en interne

Il utilise **postgresdb** comme serveur de base de données

Le nom de la base de données est : **Baserow**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: nocode**

Nocodb

```yaml
services:

  nocodb:
    image: nocodb/nocodb:latest
    container_name: nocodb
    restart: unless-stopped
    environment:
      DATABASE_USER: naomie
      DATABASE_PASSWORD: naomiep
      DATABASE_NAME: Nocodb
      DATABASE_HOST: postgresdb
      NC_PUBLIC_URL: "http://localhost:8080"
    ports:
      - "8083:8080"
    networks:
      - frontend2

volumes:
  postgres_data:

networks:
  frontend2:
    name: nocode # Use a custom name
    external: true
```

L'image utilisée est **nocodb/nocodb:latest**

Il utilise le port **8083** pour publier ces services et utilise le port **8000** en interne

Il utilise **postgresdb** comme serveur de base de données

Le nom de la base de données est : **Nocodb**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: nocode**

PIA

```yaml
version: "3"

services:
  pia-back:
    image: lrqdo/cnil-pia-back
    container_name: pia-back
    restart: always
    depends_on:
      - pia-db
    environment:
      DATABASE_USER: naomie
      DATABASE_PASSWORD: naomiep
      DATABASE_NAME: pia
      DATABASE_HOST: postgresdb
      DB_PORT: 5432
    ports:
      - "8080:3000"

  pia-front:
    image: lrqdo/cnil-pia-front
    container_name: pia-front
    restart: always
    depends_on:
      - pia-back
    environment:
      BACK_URL: http://pia-back:3000
    ports:
      - "8081:80"
    networks:
      - frontend2

volumes:
  pia-data:
networks:
  frontend2:
    name: nocode # Use a custom name
```

On a utilisé la **version 3**

L'image utilisée est **lrqdo/cnil-pia-back**

Il utilise le port **8080** pour publier ces services et utilise le port **3000** en interne

Il utilise **postgresdb** comme serveur de base de données

Le nom de la base de données est : **pia**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: nocode**

n8n

```yaml
version: "3"

services:
  n8n:
    image: docker.n8n.io/n8nio/n8n
    restart: always
    ports:
      - 87:5678
    environment:
      environment:
      DATABASE_USER: naomie
      DATABASE_PASSWORD: naomiep
      DATABASE_NAME: n8n
      DATABASE_HOST: postgresdb
    networks:
      - frontend2
    volumes:
      - n8n:/home/node/.n8n

volumes:
  n8n:
networks:
  frontend2:
    name: nocode # Use a custom name
    external: true
```

On a utilisé la **version 3**

L'image utilisée est **docker.n8n.io/n8nio/n8n**

Il utilise le port **87** pour publier ces services et utilise le port **5678** en interne

Il utilise **postgresdb** comme serveur de base de données

Le nom de la base de données est : **n8n**

Au niveau de l'environnement ce sont les identifiants de connexion

Le nom du réseau est **: nocode**

livrable3

Diagramme réseaux