# Hackathon - Ynov Toulouse 2025 : Babyfoot du futur - Cloud & Infrastructure

Équipe

* **Cloud & Infra 1** : ARROUD Rayan
* **Cloud & Infra 2** : MOUMINE Wassim

Objectif général

Le but de notre mission était de concevoir et déployer une **infrastructure complète** pour l’application web de gestion des babyfoots. Cette infrastructure devait être robuste, scalable, et permettre une gestion efficace des ressources, tout en intégrant des outils de surveillance et de sécurité adaptés pour un service utilisé par près de 1000 étudiants.

Nous avons choisi d'héberger l'infrastructure sur un **Raspberry Pi** afin de permettre au jury de tester le déploiement en situation réelle, sans nécessiter de ressources cloud externes (comme AWS, GCP, ou Azure). Le choix d'une machine physique pour héberger le service vise à offrir une approche plus tangible et un test en conditions réelles.

Table des matières

[Démarche et choix techniques 2](#_Toc211616910)

[1. Choix des technologies : 2](#_Toc211616911)

[2. Automatisation du déploiement avec Ansible : 2](#_Toc211616912)

[3. Gestion des conteneurs Docker : 2](#_Toc211616913)

[Requis fonctionnels 4](#_Toc211616914)

[1. Hébergement de l’application web : 4](#_Toc211616915)

[2. Base de données : 6](#_Toc211616916)

[3. Automatisation du déploiement : 7](#_Toc211616917)

[4. Sécurisation : 8](#_Toc211616918)

[5. Scalabilité : 9](#_Toc211616919)

[Collaboration avec les autres spécialités 9](#_Toc211616920)

[Collaboration avec l’équipe Dev FullStack 9](#_Toc211616921)

[Collaboration avec l’équipe IoT 11](#_Toc211616922)

[Collaboration avec l’équipe IA & Data 11](#_Toc211616923)

[Difficultés rencontrées 11](#_Toc211616924)

[1. Problèmes de permissions avec les volumes Docker : 11](#_Toc211616925)

[2. Déploiement sur Raspberry Pi : 11](#_Toc211616926)

[3. Compatibilité avec Docker Compose v2 : 12](#_Toc211616927)

[4. Synchronisation des données avec IoT : 12](#_Toc211616928)

[Conclusion 12](#_Toc211616929)

## Démarche et choix techniques

### Choix des technologies :

* + **Docker** : Permet de déployer facilement l’application et de gérer les services (frontend, backend, base de données).
  + **Ansible** : Outil d’automatisation utilisé pour configurer le serveur et déployer l’infrastructure en une seule commande.
  + **Nginx** : Serveur web léger utilisé pour servir les fichiers statiques (HTML, CSS, JS) et pour gérer le reverse proxy vers le backend.
  + **SQLite** : Base de données légère utilisée pour stocker les informations liées aux utilisateurs et aux réservations de babyfoots.
  + **Raspberry Pi** : Choix de la plateforme pour l’hébergement final, permettant de tester le déploiement en environnement physique.

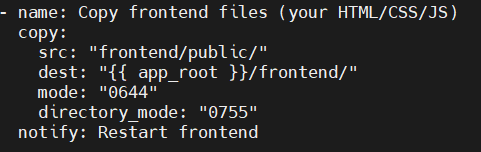
1. Automatisation du déploiement avec Ansible :

Nous avons utilisé **Ansible** pour :

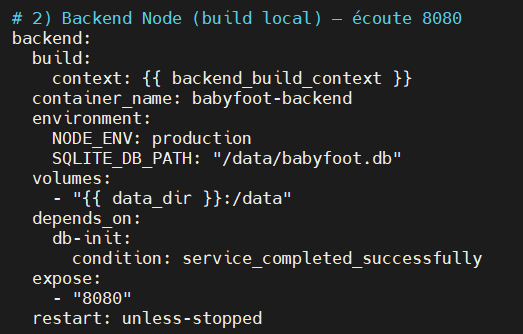
* + Déployer les services Docker : frontend, backend, et base de données.
  + Configurer Nginx pour servir les fichiers statiques et rediriger les requêtes API vers le backend.
  + Gérer les volumes Docker et s’assurer que les configurations sont appliquées correctement.

### Gestion des conteneurs Docker :

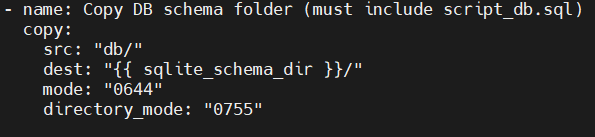
* + **Frontend** : Service Nginx pour servir les fichiers statiques.



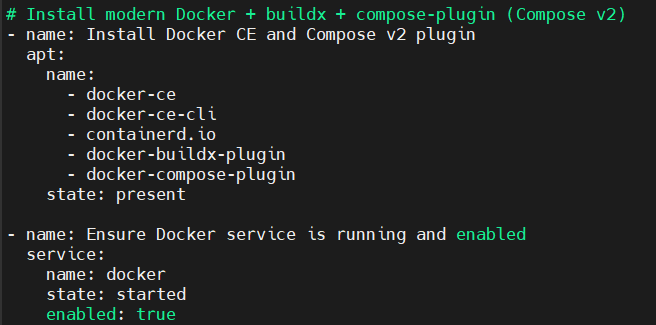
* + **Backend** : Service Node.js exposant une API pour la gestion des utilisateurs et des réservations.



* + **Base de données** : Conteneur SQLite pour stocker les données relatives aux utilisateurs et aux babyfoots.



* + L’utilisation de **Docker Compose** a permis de gérer facilement les services interconnectés et de garantir une isolation des environnements.



## Requis fonctionnels

### Hébergement de l’application web :

* + Mise en place d'un serveur **Nginx** pour servir les fichiers frontend (HTML, CSS, JS).



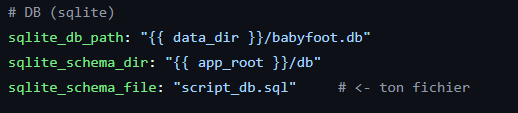
* + Déploiement d'un service **backend Node.js** pour gérer les requêtes et les interactions avec la base de données.



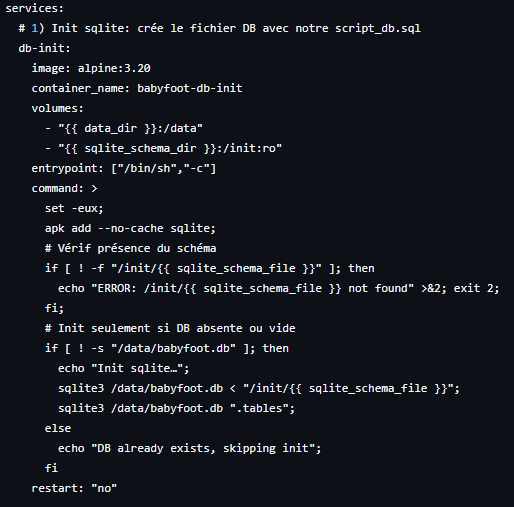
* + Utilisation de **Docker** pour encapsuler les services (frontend, backend, base de données).

### Base de données :

* + Nous avons choisi **SQLite** pour sa simplicité et son intégration facile avec le backend.

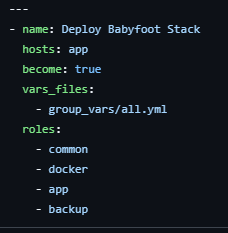


* + Une tâche d'initialisation (via Docker) est effectuée pour créer la base de données et y insérer les schémas nécessaires si la base est vide.

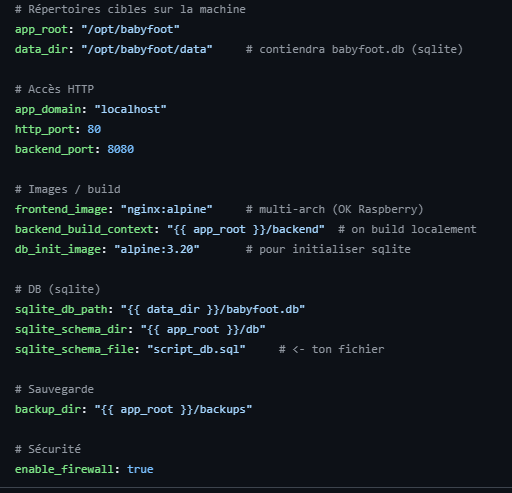


### Automatisation du déploiement :

* + Utilisation de **Ansible** pour déployer l’infrastructure en une seule commande, y compris le déploiement des services Docker et la configuration de Nginx.



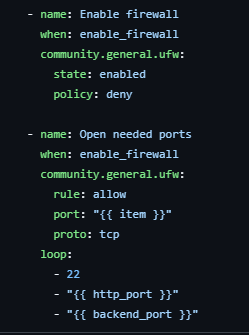
* + Scripts d’automatisation pour la gestion des services (par exemple, nettoyage, mise à jour des services, etc.).



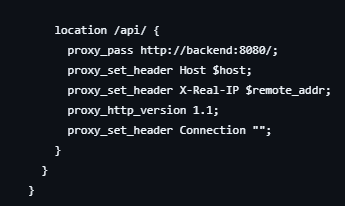
### Sécurisation :

* + Configuration de règles de sécurité de base dans **Docker** (gestion des ports et réseaux).



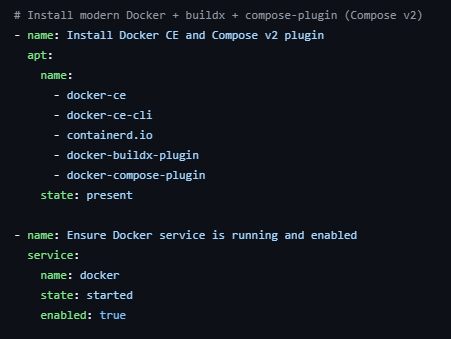


* + Reverse proxy dans **NGINX** pour plus de sécurité sur le serveur web



### Scalabilité :

* + Utilisation de Docker et de **Docker Compose** pour faciliter la mise à l’échelle de l'infrastructure en cas de besoin.



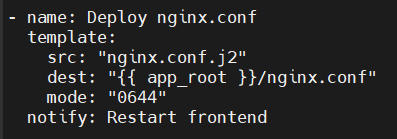
* + Pour ce qui est la mise à l’échelle automatique à travers des configurations potentielles, nous n’avons pas eu le temps d’utiliser Docker Swarm (possible à l’avenir si le projet est retenu)

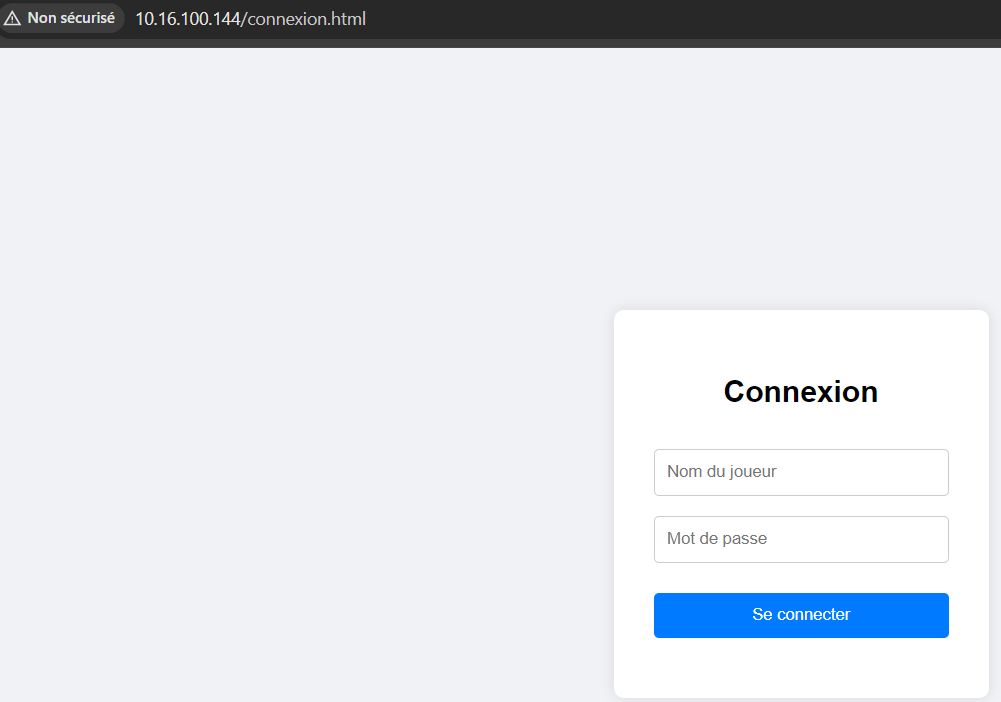
## Collaboration avec les autres spécialités

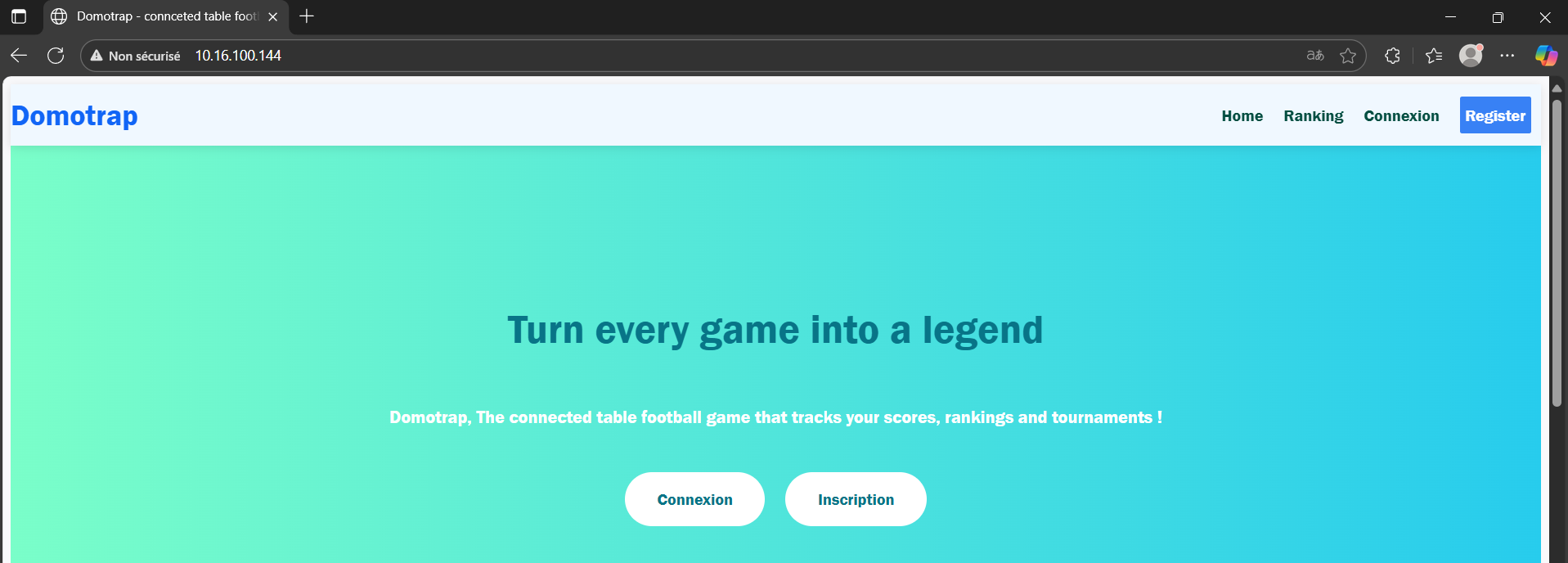
### Collaboration avec l’équipe Dev FullStack

L'équipe **Cloud & Infrastructure** a travaillé en étroite collaboration avec les équipes **Dev FullStack** pour garantir l’intégration fluide des différents services.

* **Backend Node.js** : Nous avons facilité le déploiement du backend Node.js et la configuration de la base de données SQLite. Nous avons collaboré pour déterminer la structure de la base de données et l'intégration avec les API exposées par le backend.
* **Nginx et le frontend** : Nous avons configuré Nginx pour que le frontend soit servi correctement à partir des fichiers statiques et assuré que la configuration Docker permettait de servir les fichiers à travers des volumes montés.







### Collaboration avec l’équipe IoT

L’équipe **IoT** a intégré les dispositifs connectés dans l’infrastructure. En particulier, nous avons travaillé ensemble pour récupérer les données des capteurs et les transmettre à l’application pour le suivi en temps réel des jeux.

* Nous avons intégré des services qui permettaient de gérer la communication entre le serveur backend et les capteurs, et avons mis en place un flux de données entre les babyfoots et l’application via Docker.

### Collaboration avec l’équipe IA & Data

L’équipe **IA & Data** a joué un rôle crucial dans la création de la base de données et l’analyse des données d’utilisation des babyfoots. Nous avons travaillé en étroite collaboration avec eux pour intégrer la base de données dans notre infrastructure.

* **Création de la base de données** : L’équipe IA & Data a conçu la base de données **SQLite**, en intégrant les informations relatives aux utilisateurs et aux réservations. Nous avons pris en charge la mise en place de cette base de données dans notre infrastructure Docker.
* **Traitement des données** : L’équipe IA & Data a également traité les données d’utilisation des babyfoots pour créer des statistiques utiles à l’application, ce qui a été intégré dans le backend pour fournir des informations en temps réel.
* **Collaborations API** : L’équipe IA & Data a exposé des API pour fournir des données sur l’utilisation des babyfoots, permettant au frontend et au backend d’accéder aux résultats des analyses.

## Difficultés rencontrées

1. Problèmes de permissions avec les volumes Docker :

Nous avons rencontré quelques soucis de permissions sur les volumes Docker, ce qui a entraîné des erreurs lors de la tentative de démarrage des services. Cela a été résolu en ajustant les permissions des fichiers dans le volume et en s’assurant que les conteneurs pouvaient y accéder correctement.

1. Déploiement sur Raspberry Pi :

Bien que Docker soit bien pris en charge sur Raspberry Pi, le déploiement a nécessité quelques ajustements de configuration, notamment pour assurer une gestion efficace des ressources avec une plateforme limitée en termes de RAM et de CPU.

1. Compatibilité avec Docker Compose v2 :

Le passage de Docker Compose v1 à v2 a introduit quelques changements de syntaxe, ce qui a nécessité une mise à jour de certaines tâches dans le playbook Ansible. Cela a pris un peu de temps pour adapter les commandes et faire en sorte que tout fonctionne correctement.

1. Synchronisation des données avec IoT :

Assurer que les données des capteurs IoT soient correctement transmises en temps réel à l’application a nécessité des tests et des ajustements, notamment pour la gestion de la bande passante et la fiabilité de la communication.

## Conclusion

En tant qu'équipe **Cloud & Infra**, notre rôle était de créer et maintenir l’infrastructure nécessaire pour héberger l’application Babyfoot du futur, en veillant à la mise en place des services clés, la sécurité, la scalabilité et la surveillance. Le déploiement a été réalisé en utilisant des outils d'automatisation comme **Docker**, **Ansible**, et **Nginx**.

Les principaux défis ont été de **sécuriser et déployer** les services dans un environnement physique limité (Raspberry Pi) tout en garantissant une gestion fluide des bases de données et des conteneurs. Cependant, en collaboration avec les autres équipes (Dev, IoT, IA & Data), nous avons réussi à offrir une infrastructure stable et prête à supporter l’application en production.

Nous avons assuré une **intégration fluide** entre les équipes, en particulier avec l’équipe **IA & Data**, qui a géré la création et l’analyse des données, et avec l’équipe **IoT**, pour la collecte et la transmission des données en temps réel. Ces collaborations ont permis d’enrichir l’application au maximum