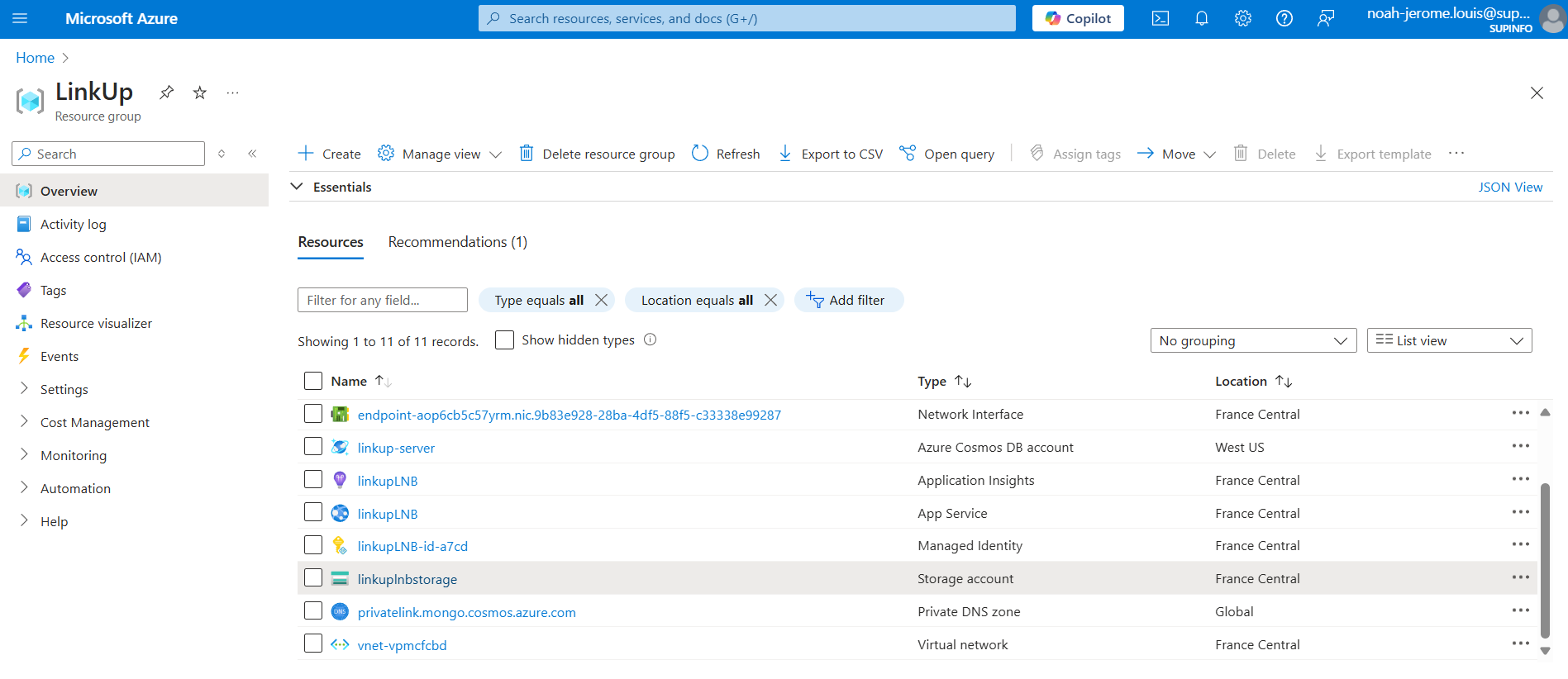
# 4DESA

# Design Considerations

Léonard Trève – Noah Louis – Benjamin Maury

## Configuration Azure pour Link Up

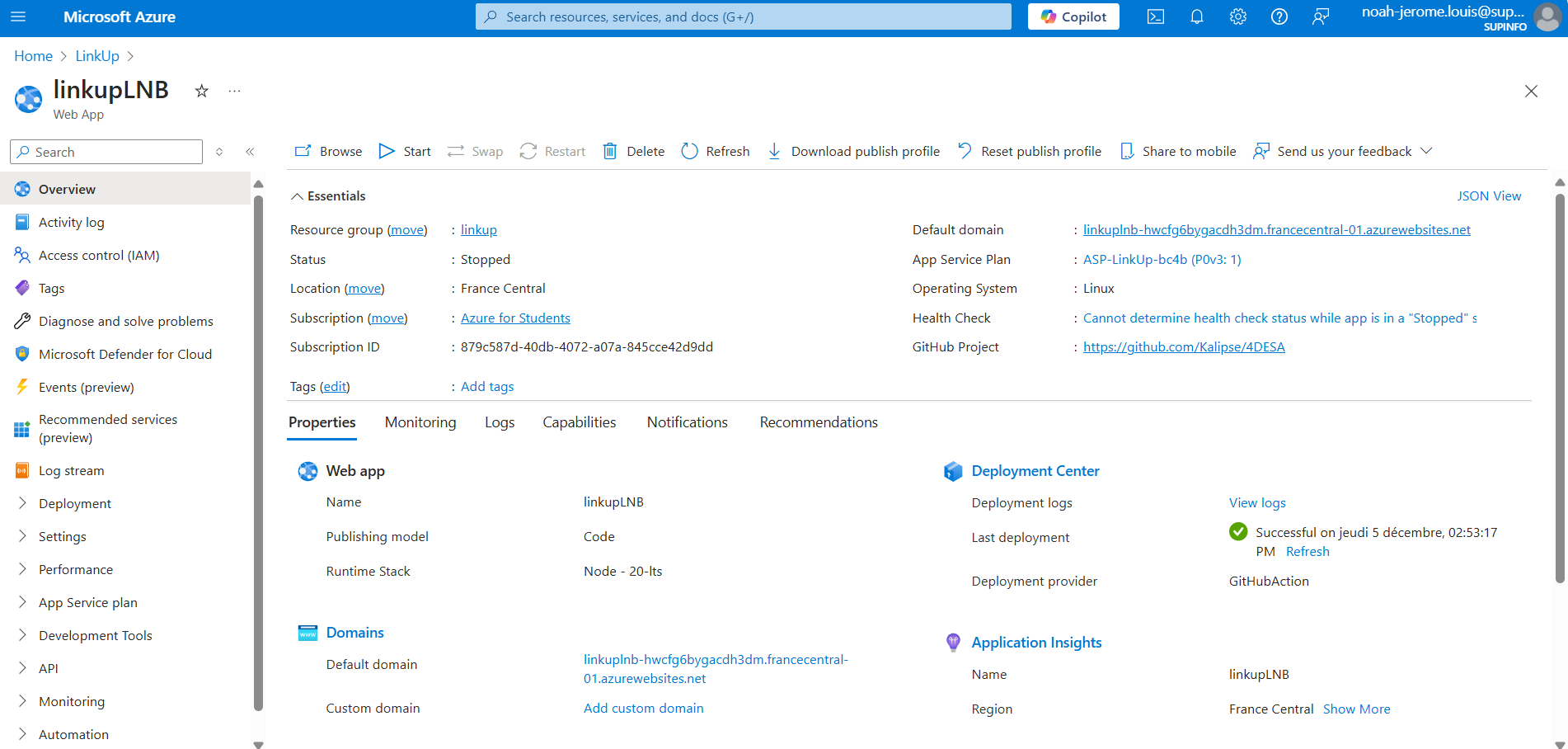
### 1. Création du groupe de ressources

****

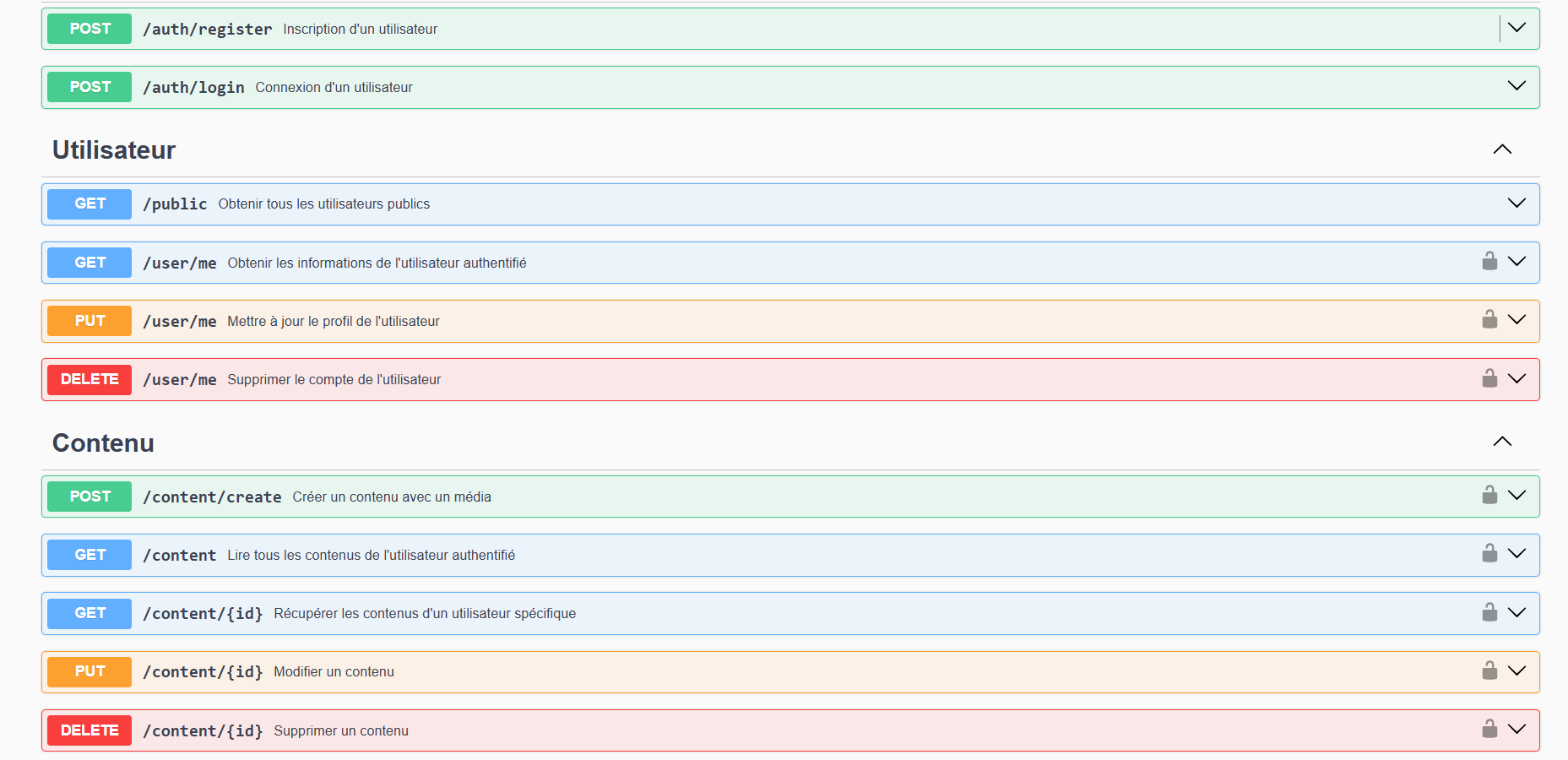
Un groupe de ressources nommé **LinkUp** a été créé pour regrouper toutes les ressources Azure nécessaires. Cela permet une gestion simplifiée et une vue d'ensemble sur les services associés

### 2. App Service : Hébergement de l'API Node.js

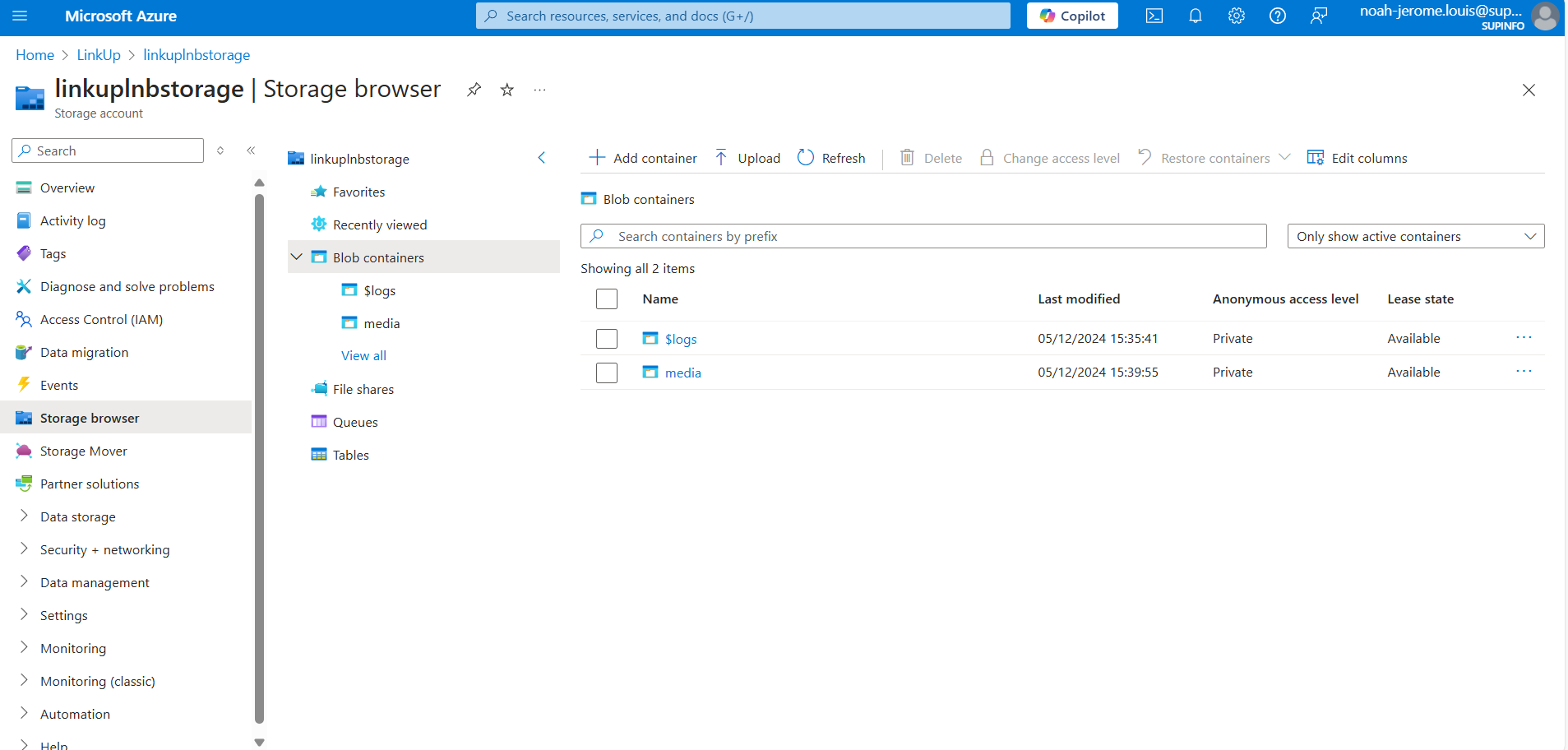
Nous avons utilisé une App Service nommée **linkupLNB** pour héberger l’API backend. Cette API a été développée en Node.js, un environnement que nous avons choisi pour sa simplicité, sa rapidité de développement, et notre expertise dans ce domaine.  
Le code source de l’API est importé directement depuis un dépôt GitHub. L'application utilise des tokens **JWT** pour gérer les authentifications et sécuriser les endpoints de **l'API**.



Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

### 3. Azure Blob Storage : Gestion des médias

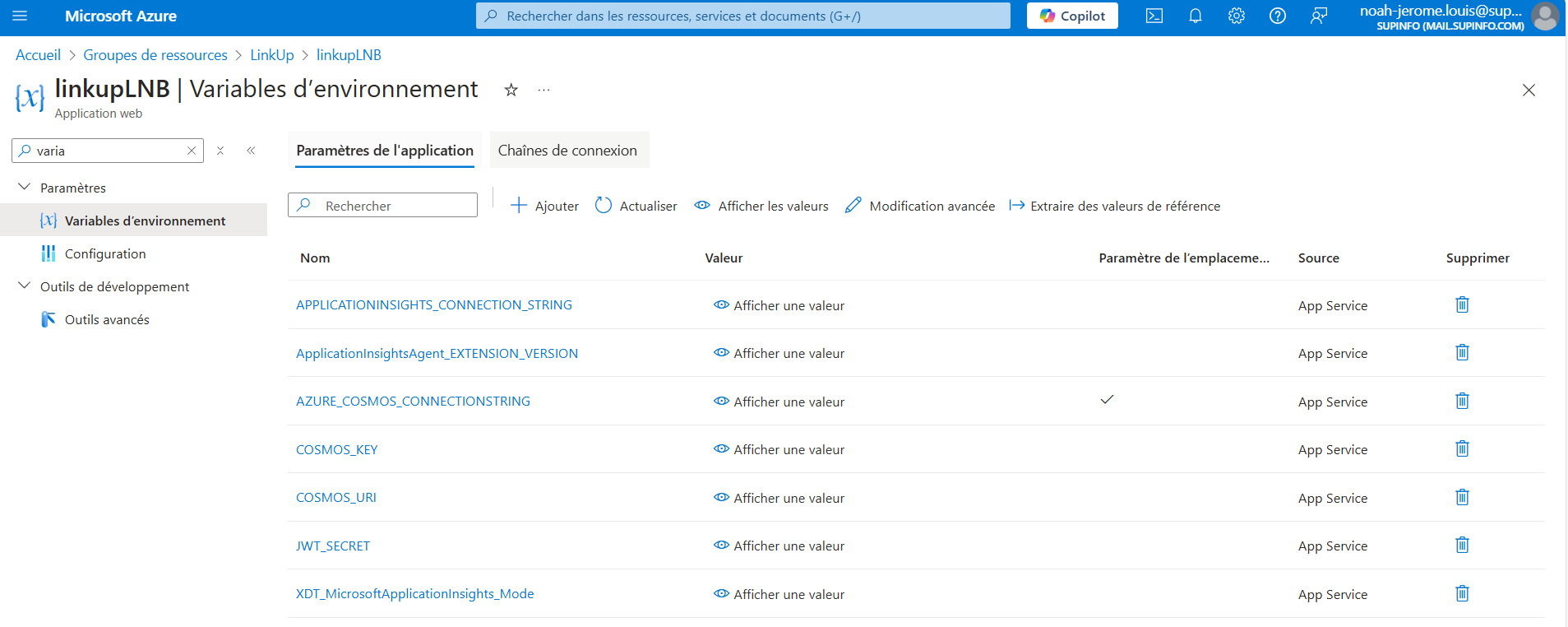
****

Pour gérer les fichiers multimédias, tels que les images et les vidéos téléchargées par les utilisateurs, nous avons configuré un Blob Storage nommé linkuplnbstorage. Les fichiers sont stockés dans un conteneur dédié appelé media.  
Nous n'avons pas appliqué de réglages avancés spécifiques à cette configuration, car les besoins actuels sont couverts par la configuration par défaut.

### Une image contenant texte, logiciel, Icône d’ordinateur, Page web Description générée automatiquement4. Azure Cosmos DB : Stockage des donnéesUne image contenant texte, capture d’écran, logiciel, Icône d’ordinateur Description générée automatiquement

Nous avons utilisé Azure Cosmos DB, en choisissant la version for NoSQL, pour gérer les données utilisateur, les contenus créés, et les métadonnées associées.   
Cosmos DB garantit également des performances constantes, même en cas d’augmentation des charges de travail.

### 5. Variables d'environnement

****

Afin de sécuriser et de connecter correctement notre application, nous avons configuré plusieurs variables d’environnement essentielles dans l’App Service. Ces variables incluent :

* COSMOS\_URI : Cette variable contient l'URI de connexion à Cosmos DB.
* COSMOS\_KEY : Cette clé est utilisée pour accéder de manière sécurisée aux services Cosmos DB.
* JWT\_SECRET : Cette clé secrète est utilisée pour signer et vérifier les tokens JWT, assurant une authentification sécurisée des utilisateurs.

### 6. Fonctionnement général

L'API backend a été développée en Node.js et utilise Cosmos DB comme système de gestion des données grâce aux variables COSMOS\_URI et COSMOS\_KEY. Elle expose des endpoints pour les principales fonctionnalités suivantes :

1. Authentification : Les utilisateurs peuvent s’inscrire et se connecter de manière sécurisée.
2. Gestion des contenus : Les utilisateurs peuvent créer, consulter, modifier et supprimer leurs contenus.

Les fichiers multimédias, tels que les images et vidéos, sont téléchargés via des endpoints sécurisés et sont stockés dans Azure Blob Storage. Chaque fichier est associé à un utilisateur et à un contenu spécifique pour garantir un accès organisé et sécurisé.

### 7. Surveillance et Scalabilité

Nous avons mis en place des mécanismes pour surveiller et ajuster les performances du système en fonction des besoins :

* **Surveillance** :  
  Nous utilisons Azure Monitor pour suivre les performances de l’App Service et des autres ressources. Des alertes sont configurées pour détecter et signaler les anomalies ou erreurs critiques.
* **Scalabilité** :  
  Les services Azure utilisés, notamment l’App Service, Cosmos DB, et Blob Storage, sont configurés pour prendre en charge une montée en charge horizontale. Cela permet d’ajouter facilement des ressources lorsque le trafic utilisateur ou le volume de données augmente.