



# **Rapport de projet Azure**

## **4<sup>ème</sup> année**

### **Ingénierie Informatique et Réseaux**

---

# **MISE EN PLACE D'UN PIPELINE CI/CD AVEC GITHUB ACTIONS ET AZURE DEVOPS POUR DEPLOYER UNE APPLICATION**

---

#### **Réalisé par :**

CHERKAOUI RHAZOINI Soufiane  
SGHIR Loubna

#### **Encadré par :**

Pr. ELHASSANI Mouhcine

# Table des matières

I. Introduction .....	3
II. Objectif du projet .....	3
III. Étapes suivies .....	3
Étape 1 : Création du projet Java avec Maven .....	3
Étape 2 : Création du dépôt GitHub et configuration Git .....	4
Étape 3 : Configuration de GitHub Actions .....	6
Étape 4 : Création des ressources sur Azure .....	7
Étape 5 : Déploiement automatique sur Azure .....	12
IV. Contraintes rencontrées lors de la réalisation du projet .....	12
V. Solution proposées .....	16
VI. Conclusion .....	19

---

## Introduction

Dans le cadre de notre projet de fin de semestre, nous avons choisi de mettre en œuvre une démarche DevOps complète afin d'automatiser l'intégration et le déploiement d'une application Java simple sur la plateforme cloud Microsoft Azure. L'objectif principal est d'illustrer comment utiliser **GitHub Actions** pour l'intégration continue (CI) et **Azure App Service** pour le déploiement continu (CD), tout en intégrant des bonnes pratiques telles que le versionnage, la conteneurisation et la livraison rapide.

---

### I. Introduction

Dans le cadre de notre projet de fin de semestre, nous avons choisi de mettre en œuvre une démarche DevOps complète afin d'automatiser l'intégration et le déploiement d'une application Java simple sur la plateforme cloud Microsoft Azure. L'objectif principal est d'illustrer comment utiliser **GitHub Actions** pour l'intégration continue (CI) et **Azure App Service** pour le déploiement continu (CD), tout en intégrant des bonnes pratiques telles que le visionnage, la conteneurisation et la livraison rapide.

---

### II. Objectif du projet

- Créer une application Java simple avec **Maven**.
  - Automatiser la compilation, le test et la construction avec GitHub Actions.
  - Déployer automatiquement l'application sur Azure à chaque mise à jour du code (push/pull request).
  - Explorer l'interaction entre GitHub et les services cloud (Azure DevOps, Azure App Service).
  - Fournir un pipeline CI/CD fonctionnel et extensible pour de futurs projets.
- 

### III. Étapes suivies

#### Étape 1 : Création du projet Java avec Maven

```
Java-Azure-App/  
├─ src/  
├─ pom.xml  
└─ .github/  
    └─ workflows/  
        └─ ci.yml  <-- Ton fichier GitHub Actions
```

- Génération d'un projet Java minimal avec un fichier `pom.xml`.
- Ajout de la classe principale `App.java`.



```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4       http://maven.apache.org/xsd/maven-4.0.0.xsd">
5
6     <modelVersion>4.0.0</modelVersion>
7     <groupId>com.javaazureapp</groupId>
8     <artifactId>java-azure-app</artifactId>
9     <version>1.0-SNAPSHOT</version>
10
11     <properties>
12       <maven.compiler.source>23</maven.compiler.source>
13       <maven.compiler.target>23</maven.compiler.target>
14       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     </properties>
16

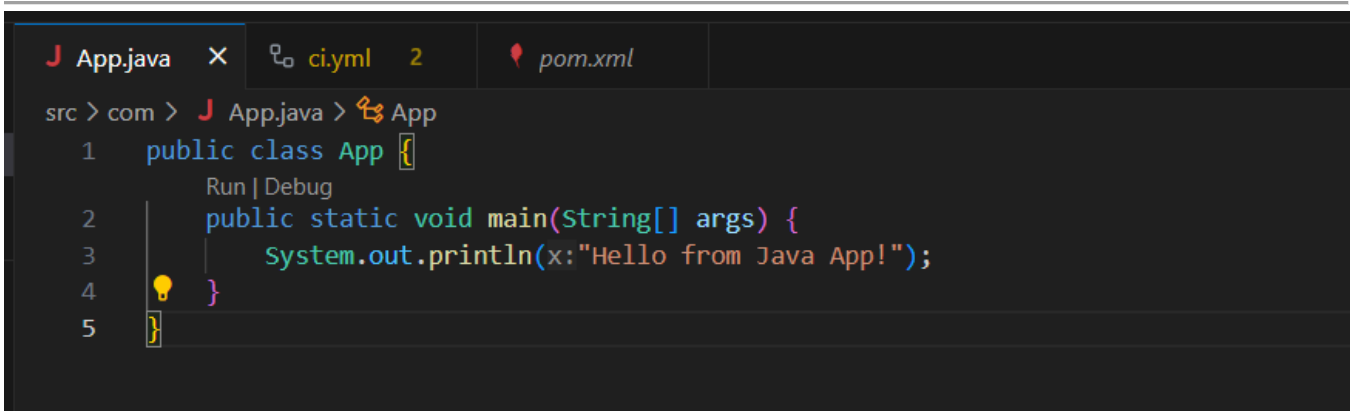
```

- Configuration du `pom.xml` pour compiler le projet et générer un `.jar` exécutable.

```

Java-Azure-App/
├─ src/
├─ pom.xml
├─ .github/
│   └─ workflows/
│       └─ ci.yml  <-- Ton fichier GitHub Actions

```



```


src > com > J App.java > App
1 public class App {
2     Run | Debug
3     public static void main(String[] args) {
4         System.out.println(x:"Hello from Java App!");
5     }
6 }

```

## Étape 2 : Création du dépôt GitHub et configuration Git

- Initialisation du dépôt Git localement.
- Ajout du dépôt distant GitHub.
- Résolution de conflits liés à l'origine du projet Git.

- Push vers la branche principale `main`.



Java-azure-app

Public

Pin

Unwatch 1

main


1 Branch

0 Tags

Go to file

+

Code








soufianecherk

deployment




ed11abc · 11 hours ago

7 Commits

 .github/workflows	deployment	11 hours ago
 .vscode	Fix	15 hours ago
 src/com	Quick fix	14 hours ago
 target	deployment	11 hours ago
 pom.xml	Quick fix	14 hours ago

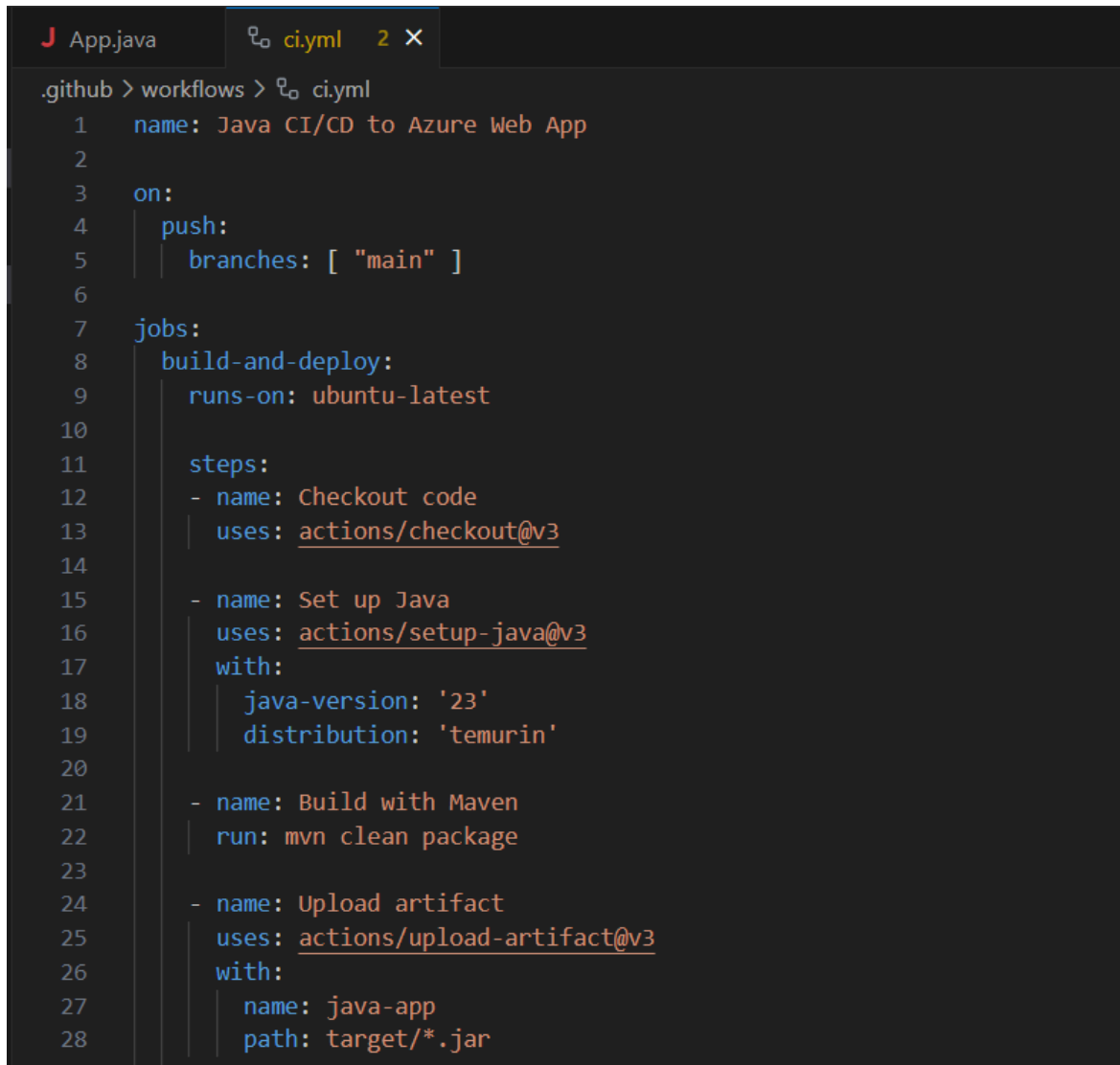
 [Insérer capture d'écran du dépôt GitHub et terminal avec push réussi]

- Dossier src/com

Name	Last commit message	Last commit date
 ..		
 App.class	Quick fix	16 hours ago
 App.java	Quick fix	16 hours ago

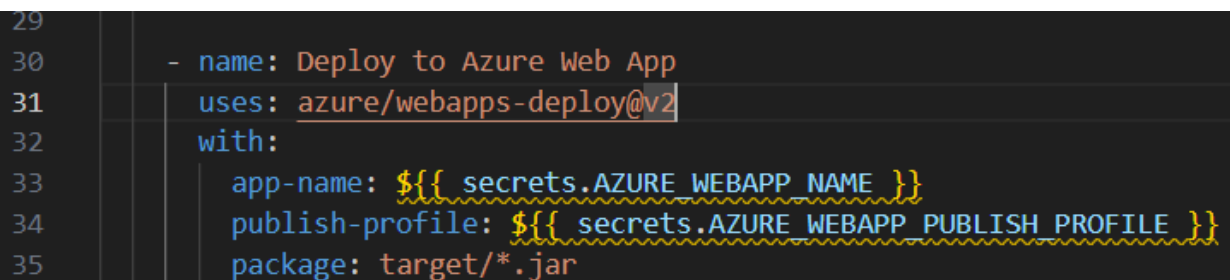
### Étape 3 : Configuration de GitHub Actions

- Création d'un dossier `.github/workflows` et ajout du fichier `ci.yml`.
- Définition du pipeline GitHub Actions pour compiler le projet avec Maven.



```
App.java  ci.yml 2 x
.github > workflows > ci.yml
1  name: Java CI/CD to Azure Web App
2
3  on:
4    push:
5      branches: [ "main" ]
6
7  jobs:
8    build-and-deploy:
9      runs-on: ubuntu-latest
10
11     steps:
12       - name: Checkout code
13         uses: actions/checkout@v3
14
15       - name: Set up Java
16         uses: actions/setup-java@v3
17         with:
18           java-version: '23'
19           distribution: 'temurin'
20
21       - name: Build with Maven
22         run: mvn clean package
23
24       - name: Upload artifact
25         uses: actions/upload-artifact@v3
26         with:
27           name: java-app
28           path: target/*.jar
29
```

- Ajout des étapes pour tester le code et préparer le déploiement.



```
29
30     - name: Deploy to Azure Web App
31       uses: azure/webapps-deploy@v2
32       with:
33         app-name: ${ secrets.AZURE_WEBAPP_NAME }
34         publish-profile: ${ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }
35         package: target/*.jar
```

📷 [Insérer capture d'écran du fichier ci.yml et des exécutions GitHub Actions]

## Étape 4 : Création des ressources sur Azure

- Création d'un groupe de ressources.
- Création d'un **App Service Plan (Linux)**.
- Création d'une **Web App Java** sur Azure configurée pour Maven.
- Liaison de GitHub à la Web App pour activer le déploiement continu.

📸 [Insérer capture d'écran du portail Azure avec les ressources créées]

**Objectif :** Créer une Web App Java sur Azure pour déploiement CI/CD

### Ressources

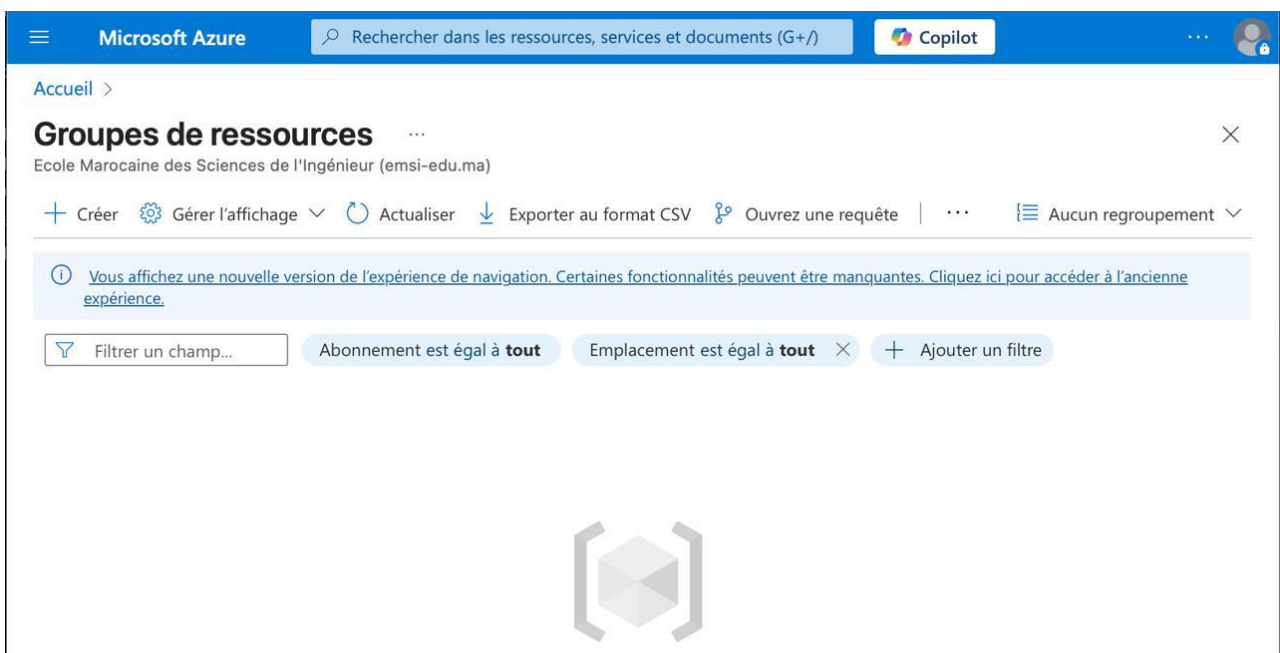
Récent Favori

Nom	Type	Dernier affichage
 Java-Azure-App	App Service	il y a 26 minutes
 Java-App-RG	Groupe de ressources	il y a 2 heures
 Java-App-Plan	Plan App Service	il y a 2 heures

Tout afficher

### 1. Créer un Groupe de ressources

Un groupe de ressources regroupe tous les éléments liés à ton projet.

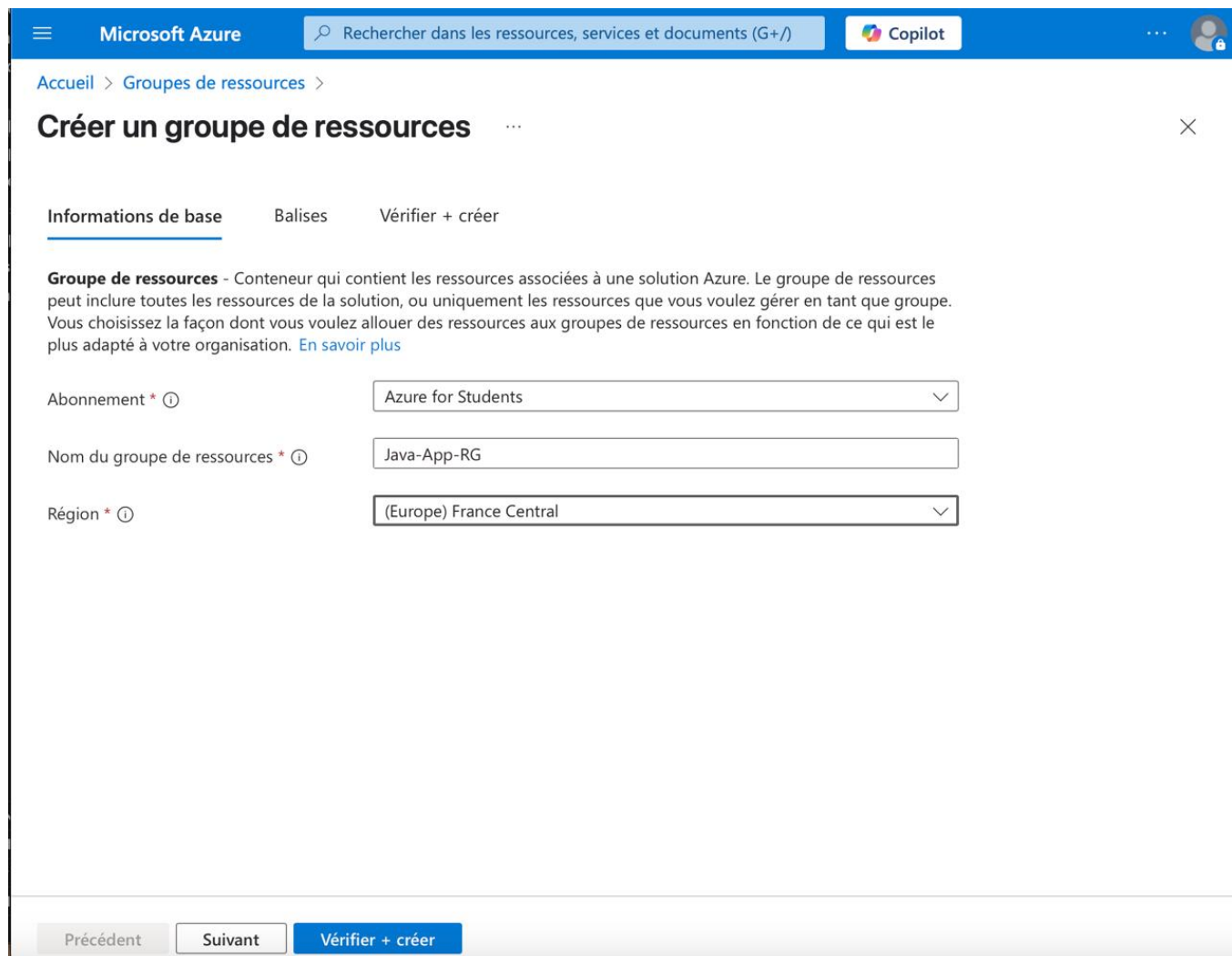


## Renseigne

Nom : java-app-rg (ou autre)

Région : Choisissez celle proche de toi (ex : "France Central")

Clique sur "Revoir + Créer", puis "Créer"



The screenshot shows the 'Créer un groupe de ressources' (Create a resource group) page in the Microsoft Azure portal. The page has a blue header with the 'Microsoft Azure' logo, a search bar, and a 'Copilot' button. The breadcrumb trail is 'Accueil > Groupes de ressources >'. The main title is 'Créer un groupe de ressources' with a close button (X) on the right. Below the title are three tabs: 'Informations de base' (selected), 'Balises', and 'Vérifier + créer'. A descriptive paragraph explains that a resource group is a container for Azure resources. Below this, there are three form fields: 'Abonnement \*' with a dropdown menu showing 'Azure for Students', 'Nom du groupe de ressources \*' with a text input field containing 'Java-App-RG', and 'Région \*' with a dropdown menu showing '(Europe) France Central'. At the bottom, there are three buttons: 'Précédent' (disabled), 'Suivant' (disabled), and 'Vérifier + créer' (active).

## 2. Créer un App Service Plan

C'est la machine virtuelle partagée qui héberge ta Web App.

Recherchez "App Service plans" dans le portail Azure

Clique sur "Créer"



Microsoft Azure

Rechercher dans les ressources, services et documents (G+/)

Copilot

Accueil >

## Plans App Service

Ecole Marocaine des Sciences de l'Ingénieur (emsi-edu.ma)

+ Créer ⚙️ Gérer la vue Actualiser ⬇️ Exporter au format CSV 🔗 Ouvrir une requête | 🏷️ Attribuer des étiquettes

Filtrer un champ... Abonnement égal à **tout** Groupe de ressources égal à **tout** ✕ + Ajouter un filtre Plus (1)

Affichage de 0 à 0 sur 0 enregistrements. Aucun regroupement Vue liste

Nom ↑↓

### Renseigne :

**Groupe de ressources :** Sélectionne java-app-rg

**Système d'exploitation :** Linux

**Région :** identique au groupe

Clique sur "**Revoir + Créer**", puis "Créer"

Microsoft Azure

Rechercher dans les ressources, services et documents (G+/)

Copilot

Accueil > Plans App Service >

## Créer un plan App Service

Détails du projet

Sélectionnez un abonnement pour gérer les coûts et les ressources déployées. Utilisez les groupes de ressources comme des dossiers pour organiser et gérer toutes vos ressources.

Abonnement \* Azure for Students

Groupe de ressources \* Java-App-RG  
[Créer nouveau](#)

Détails du plan App Service

Nom \* Java-App-Plan ✓

Système d'exploitation \* ☒ Linux ☐ Windows

Région \* France Central

Niveau tarifaire

Le niveau tarifaire du plan App Service détermine l'emplacement, les fonctionnalités, le coût et les ressources de calcul associés à votre application. [En savoir plus](#)

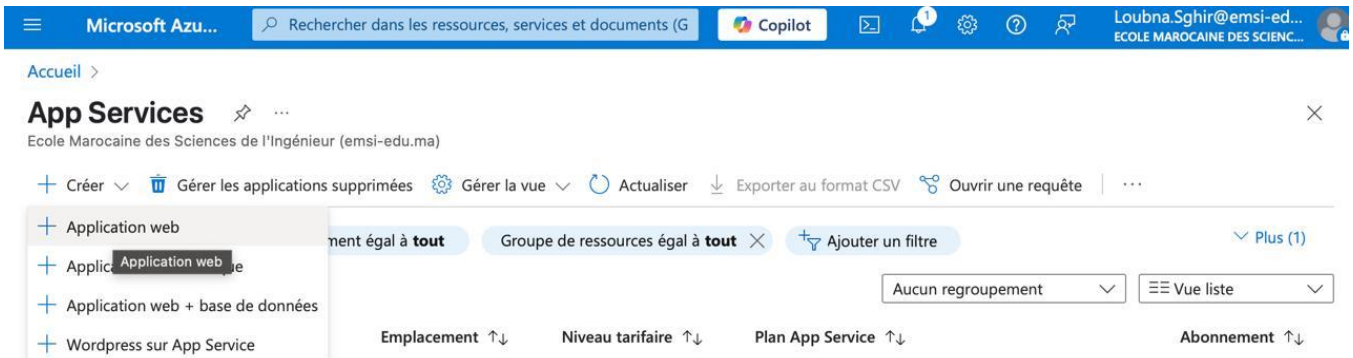
Plan de tarification Gratuit F1 (Infrastructure partagée)  
[Découvrir les plans de tarification](#)

Redondance de zone

Vérifier + créer < Précédent Suivant : Balises >

### 3. Créer la Web App Java

- C'est ici que notre application sera déployée !



#### Renseigne :

**Nom de l'application :** java-azure-app (ou autre dispo)

**Groupe de ressources :** java-app-rg

**Plan App Service :** java-app-plan

**Runtime stack :** Java 21

**Version de Java :** Java 21 (LTS)

**Stack web :** Java SE

Clique sur "**Revoir + Créer**", puis "**Créer**"

A screenshot of the 'Créer une application web' (Create a web application) form in the Microsoft Azure portal. The form is filled with the following values: Abonnement: Azure for Students, Groupe de ressources: Java-App-RG, Nom: Java-Azure-App, Publier: Code, Pile d'exécution: Java 21, Pile serveur web Java: Java SE (Embedded Web Server), Système d'exploitation: Linux, Région: France Central. The 'Vérifier + créer' button is highlighted. The form also includes a section for 'Détails de l'instance' (Instance details) and a note about securing the host name.

[Accueil](#) > [App Services](#) >

## Créer une application web ...

Publier	Code
Pile d'exécution	Java 21
Pile serveur web Java	Java SE (Embedded Web Server)

### Plan App Service

Nom	Java-App-Plan
Système d'exploitation	Linux
Région	France Central
Référence	Gratuit
ACU	Infrastructure partagée
Mémoire	1 Go de mémoire

### Superviser + sécuriser (nouveau)

Application Insights	Activé
Nom	Java-Azure-App
Région	France Central

### Déploiement

Authentification de base	Désactivé
Déploiement continu	Non activé/configuré après la création de l'application

Créer

< Précédent

Suivant >

[Télécharger un modèle pour automatisation](#)

✦ Une fois créé, on vérifie que tu peux accéder à l'URL <https://<ton-nom>.azurewebsites.net>

^ Bases

Vue JSON

Groupe de ressour... : [java-app-rg](#)

État : Exécution

Emplacement (... : France Central

Abonnement ( : [java-azure-app-cscmebgzehacguec.francecentral-01.azurewebsites.net](#)

ID d'abonnem

Domaine par défaut : [java-azure-app-cscmebgzehacguec.francecentral...](#)

Plan App Service : Java-App-Plan (F1: 1)

Système d'exploita... : Linux

Vérification de l'int... : Non configuré

Étiquettes ([modifier](#)) : [Ajouter des étiquettes](#)


Propriétés

Supervision

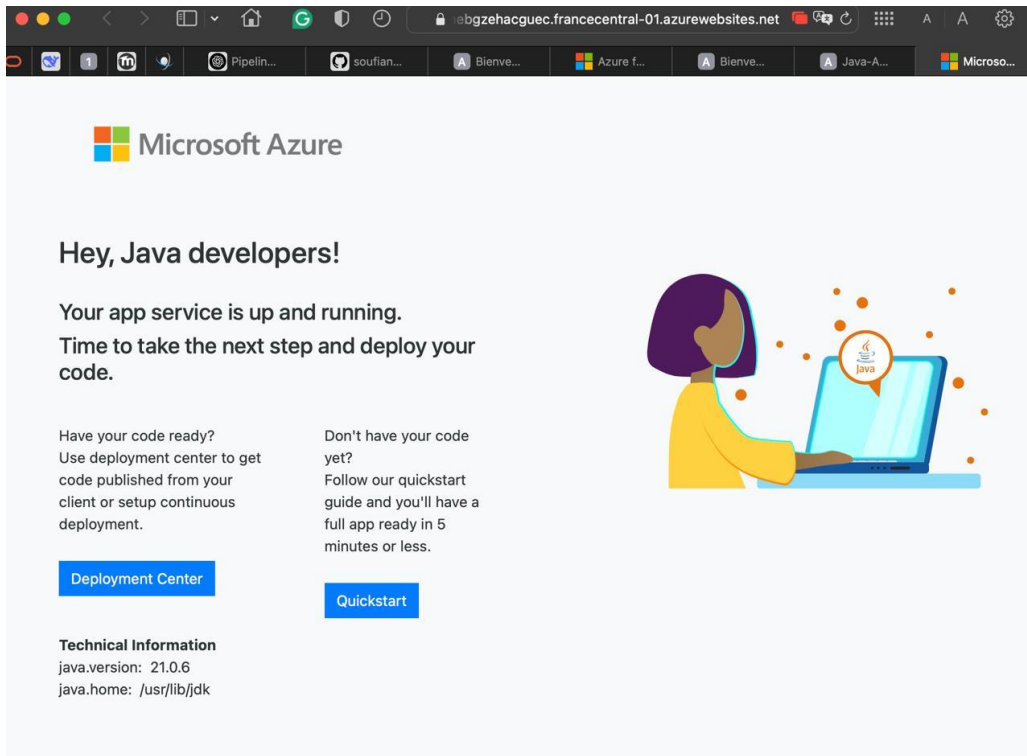
Journaux

Fonctionnalités

Notifications (0)

 Application web

Nom



#### 4. Télécharger le Profil de publication

Dans ta Web App créée (java-azure-app), va dans "Centre de déploiement" ou "Vue d'ensemble"

- Clique sur "Obtenir le profil de publication"
- Un fichier .PublishSettings sera téléchargé
- Ouvre-le, copie tout le contenu
- Va dans ton dépôt GitHub :

**Paramètres > Secrets and Variables > Actions**

Crée un secret :

**Nom : AZURE\_WEBAPP\_PUBLISH\_PROFILE**

Crée aussi un autre secret :

**Nom : AZURE\_WEBAPP\_NAME**

---

### Étape 5 : Déploiement automatique sur Azure

- Push sur la branche `main`.
- Exécution du pipeline CI/CD via GitHub Actions.

- Déploiement automatique vers Azure App Service sans Docker (utilisation du code source + Maven sur Azure).
- Vérification en ligne de l'application déployée.

Accueil > App Services > Java-Azure-App

## Java-Azure-App | Centre de déploiement

Application web

Rechercher

Connecté en tant que 02loubna-sghir [Changer de compte](#)

Organisation \* 02loubna-sghir

Dépôt \* ① Java-azure-app

Branche \* main

Option de workflow \*

☒ Ajouter un workflow : ajouter un nouveau fichier de workflow main\_Java-Azure-App.yml dans le dépôt et la branche sélectionnés.

☐ Utiliser le workflow disponible : utiliser l'un des fichiers de workflow disponibles dans le dépôt et la branche sélectionnés.

### Informations sur la build

Pile de runtime	Java
Version d'exécution	21.0
Pile serveur web Java	Java SE

### Paramètres d'authentification

Sélectionnez le mode d'authentification de votre flux de travail GitHub Actions auprès d'Azure. Si vous choisissez l'identité affectée par l'utilisateur, l'identité sélectionnée sera fédérée avec GitHub en tant que client autorisé et recevra des droits d'écriture sur l'application. [En savoir plus](#)

Type d'authentification \* ☒ Identité affectée par l'utilisateur

📸 [Insérer capture d'écran de la Web App en ligne + logs GitHub Actions]

## ✓ Conclusion

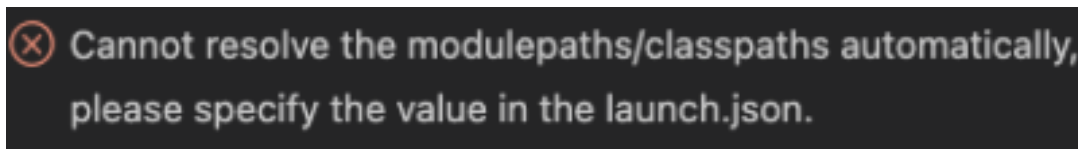
Ce projet nous a permis de découvrir et de mettre en œuvre les concepts fondamentaux du DevOps, notamment l'automatisation des processus de build et de déploiement. Grâce à **GitHub Actions**, nous avons pu configurer une chaîne CI/CD fiable qui déploie automatiquement une application Java sur **Azure App Service**. Ce workflow est facilement adaptable à d'autres types d'applications (Spring Boot, Node.js, Python...) et peut évoluer vers des architectures plus complexes comme **Azure Kubernetes Service (AKS)** ou **Docker avec Azure Container Registry**.

Souhaites-tu que je te génère aussi une version Word prête à remplir avec les captures d'écran ?

## IV. Contraintes rencontrées lors de la réalisation du projet

### 1. Problèmes de configuration du pipeline CI/CD

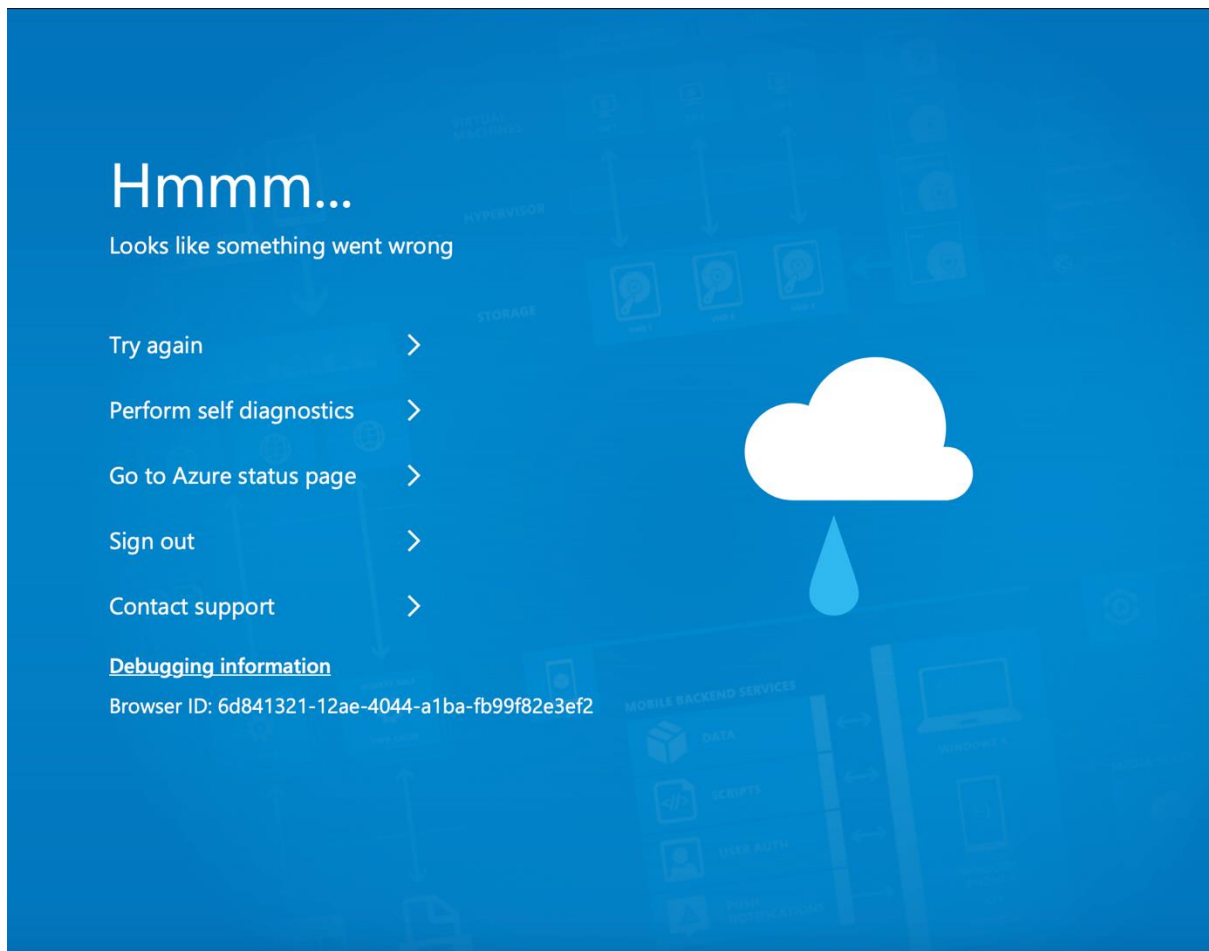
Lors de la mise en place du pipeline CI/CD avec GitHub Actions pour notre application Java basée sur Maven, nous avons rencontré des difficultés techniques. Une des premières contraintes concernait la configuration du fichier `launch.json`, qui affichait l'erreur "**Cannot resolve the modulepaths/classpaths automatically**". Cette erreur nécessitait une configuration manuelle des chemins de classes (classpath), ce qui a nécessité des recherches supplémentaires et une adaptation à la structure de notre projet.



### 2. Panne temporaire du portail Azure







- Lors de la configuration du déploiement et des tests sur le portail Azure, nous avons été confrontés à une erreur bloquante affichant le message :

*"Hmmm... Looks like something went wrong"*

Cette erreur apparaissait de manière aléatoire, généralement lors de la connexion ou du chargement de certaines ressources. Elle empêchait l'accès aux services nécessaires (App Services, tableau de bord de déploiement, etc.).

Nous avons tenté plusieurs actions comme :

- Recharger la page,
- Réinitialiser la session en se déconnectant puis reconnectant,
- Vérifier l'état des services via la **Azure Status Page**.

Il s'agissait probablement d'un dysfonctionnement temporaire ou d'un problème réseau. Cette situation a occasionné une **perte de temps** dans la phase de déploiement et nécessité une certaine **adaptabilité** pour continuer l'avancement du projet.

### 3. Déploiement automatique sur Azure

Nous avons ensuite mis en place un déploiement automatique vers **Azure App Service**. Bien que GitHub propose un fichier de workflow généré automatiquement, ce dernier ne correspondait pas entièrement aux spécificités de notre projet.

```
main_java-azure-app.yml .github/workflows 3
⚠ Context access might be invalid: AZUREAPPSERVICE_CLIENTID_9DF9844FEE1E4AC98FD743CA856F3AA9 [Ln 55, Col 22]
⚠ Context access might be invalid: AZUREAPPSERVICE_TENANTID_FFB575D0C9934EDEAF414D6A11BD5AD4 [Ln 56, Col 22]
⚠ Context access might be invalid: AZUREAPPSERVICE_SUBSCRIPTIONID_6BC8F7D2BB1E475EA619D342DE68E689 [Ln 57, Col 28]
```

## 4. Problèmes de déploiement :

Paramètres    Conteneurs (nouveau)    Journaux    Informations d'identification FTPS

🔄 Actualiser    🗑 Supprimer

ⓘ Aucun déploiement trouvé. Si vous venez de configurer CI/CD, actualisez les journaux pour rechercher les derniers déploiements.

Bien que nous ayons mis en place un pipeline CI/CD via GitHub Actions, l'étape finale de **déploiement automatique sur Azure App Service** n'a pas pu être achevée. En accédant à l'onglet *Journaux* dans le portail Azure, aucun déploiement n'a été détecté malgré plusieurs essais et reconfigurations. Cette situation peut s'expliquer par :

- Un problème de configuration dans le fichier de workflow YAML,
- Un défaut de liaison entre GitHub et Azure,
- Ou encore une erreur de permissions liée aux secrets Azure non reconnus.

Ce blocage nous a empêchés de **valider le déploiement automatique**, ce qui constitue une contrainte majeure dans l'automatisation complète du cycle DevOps. Pour aller plus loin, il aurait été pertinent de procéder à un déploiement manuel temporaire.

## V. Solutions proposées

### 1. Problèmes de configuration du pipeline CI/CD

#### Contrainte :

Lors de la mise en place du pipeline CI/CD avec GitHub Actions pour notre application Java (basée sur Maven), nous avons rencontré l'erreur :

**"Cannot resolve the modulepaths/classpaths automatically"**

Cette erreur était liée à une mauvaise détection automatique du classpath dans le fichier launch.json.

#### Solutions proposées :



- **Configurer manuellement le classpath** : en utilisant une structure de projet bien définie (src/main/java pour les sources, target/classes pour les classes compilées), nous avons explicitement renseigné le chemin vers le dossier target/classes dans la configuration.
  - **Consulter la documentation officielle de VS Code et Maven** pour adapter le launch.json au projet.
  - **Installer les extensions nécessaires** dans VS Code telles que *Java Extension Pack* pour une meilleure gestion du classpath.
- 

## 2. Panne temporaire du portail Azure

### Contrainte :

Le portail Azure affichait le message aléatoire :

"Hmmm... Looks like something went wrong"

Rendant l'accès aux services (App Services, Dashboard, etc.) instable.

### Solutions proposées :

- **Rechargement manuel et reconnexion** : pour forcer le rafraîchissement de la session.
  - **Vérification de l'état des services Azure** via [status.azure.com](https://status.azure.com).
  - **Changer de navigateur ou de réseau** : parfois le problème est local (cache, cookies, DNS).
  - **Planification des déploiements** en dehors des heures de maintenance ou de forte affluence pour minimiser l'impact.
- 

## 3. Déploiement automatique sur Azure (workflow GitHub Actions non adapté)

### Contrainte :

Le fichier yaml généré automatiquement par GitHub Actions ne correspondait pas parfaitement aux besoins spécifiques de notre projet Maven.

### Solutions proposées :

- **Modifier manuellement le workflow** : ajustement des étapes de compilation et de déploiement en ajoutant :
  - **Utiliser les secrets GitHub** (AZURE\_WEBAPP\_PUBLISH\_PROFILE) correctement configurés dans le repository.
  - **Tester localement le JAR généré** (target/app.jar) pour s'assurer qu'il est bien exécutable.
-

## 4. Problèmes de déploiement non détecté sur Azure

### Contrainte :

Malgré la configuration du pipeline, aucun déploiement n'était détecté dans les journaux Azure. Cela a entravé la validation du pipeline.

### Solutions proposées :

- **Vérifier les secrets GitHub** : s'assurer que les clés comme `AZURE_WEBAPP_PUBLISH_PROFILE` sont valides, non expirées, et bien nommées.
  - **Recréer le lien entre GitHub et Azure** via le portail Azure (section *Deployment Center* > GitHub > Reconnect).
  - **Analyser les journaux GitHub Actions** pour détecter les erreurs dans les étapes build ou deploy.
  - **Effectuer un déploiement manuel temporaire** pour valider la fonctionnalité avant de retenter l'automatisation.
  - **Activer le diagnostic Azure** pour obtenir des logs détaillés sur les tentatives de déploiement.
-

## VI. Conclusion

Ce projet nous a permis de mettre en pratique les concepts fondamentaux du développement Java tout en intégrant des outils modernes de gestion de code et de déploiement, tels que Maven, GitHub Actions et Azure App Service.

À travers la création d'une application Java complète, puis son intégration dans un pipeline CI/CD, nous avons non seulement consolidé nos compétences en développement logiciel, mais aussi exploré les bonnes pratiques du DevOps. La configuration de l'automatisation du build, des tests et du déploiement nous a offert une vision plus globale du cycle de vie d'une application.

Malgré les contraintes rencontrées, comme la configuration complexe du pipeline ou les interruptions du portail Azure, nous avons su faire preuve de persévérance et d'adaptabilité. Ces défis ont renforcé notre capacité à diagnostiquer des erreurs, à ajuster les configurations et à rechercher des solutions pertinentes.

En somme, ce projet a été une expérience formatrice qui nous a permis :

- D'approfondir nos compétences techniques,
- De mieux comprendre les enjeux de l'automatisation dans le développement moderne,
- Et de travailler de manière structurée avec des outils professionnels utilisés en entreprise.

Il constitue une base solide pour aborder des projets plus complexes à l'avenir, et pour évoluer vers des pratiques DevOps plus avancées.

---