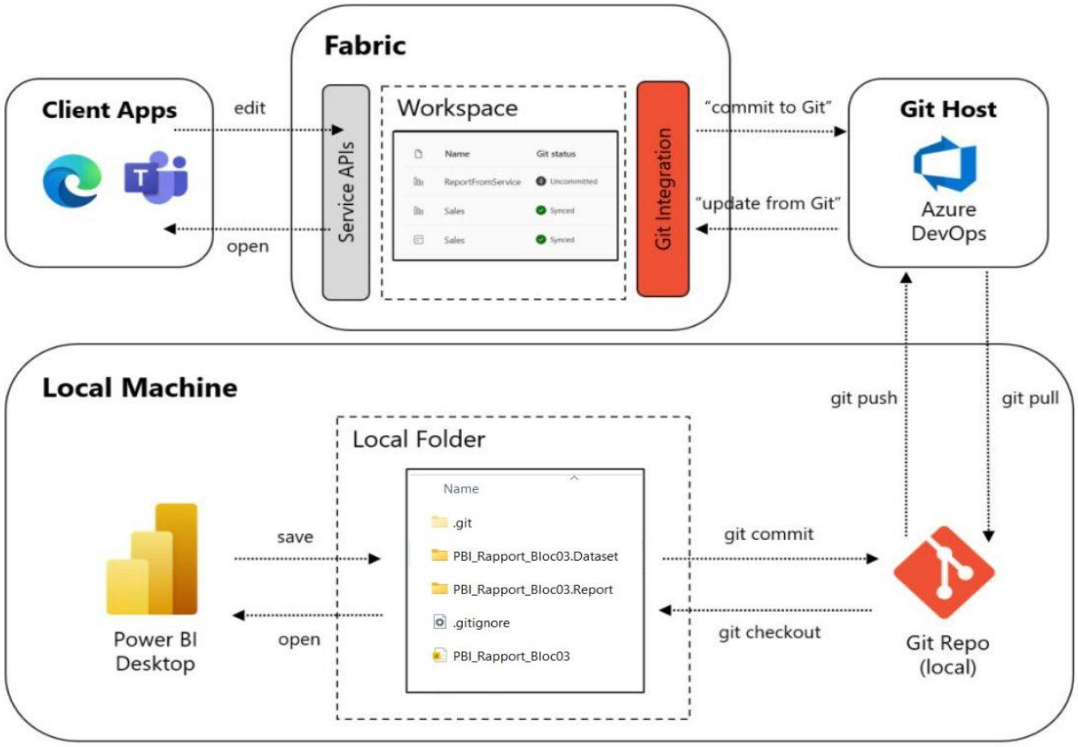
Approche générale à propos Test unitaire et fonctionnel pour notre Application Power BI

Pour PowerBI vu ses spécificités et qu'il n'est pas un langage de programmation traditionnel les test unitaire et fonctionnels sont intégrés dans un Pipeline CI/CD (Continuous Integration / Continious Deployement) dans Aure DevOps. Pour décrire mieux notre flux de travail le diagram dans la photo ci-dessous décrit bien la relation entre notre projet Power BI, Azure DevOps et Microsoft Fabric autrement c'est le mode de **le Mode Développeur de Power BI Desktop et l'Intégration Fabric Git ;**



le Mode Développeur de Power BI Desktop et l'Intégration Fabric Git

Alors qu'est-ce que : Azure DevOps, integration Fabric et à quoi servent dans notre projet ;

Définition :

Azure DevOps est un service cloud qui fournit un ensemble d'outils et de fonctionnalités pour le développement et le déploiement de logiciels. Il comprend des fonctionnalités telles que :

- Gestion des versions
- Intégration continue (CI) et déploiement continu (CD)
- Gestion des builds

- Gestion des tests

Azure DevOps peut être utilisé avec Fabric Power BI pour automatiser le cycle de vie du développement et du déploiement des rapports Power BI. Cela peut aider à améliorer la qualité des rapports, à réduire les risques et à accélérer le temps de mise sur le marché.

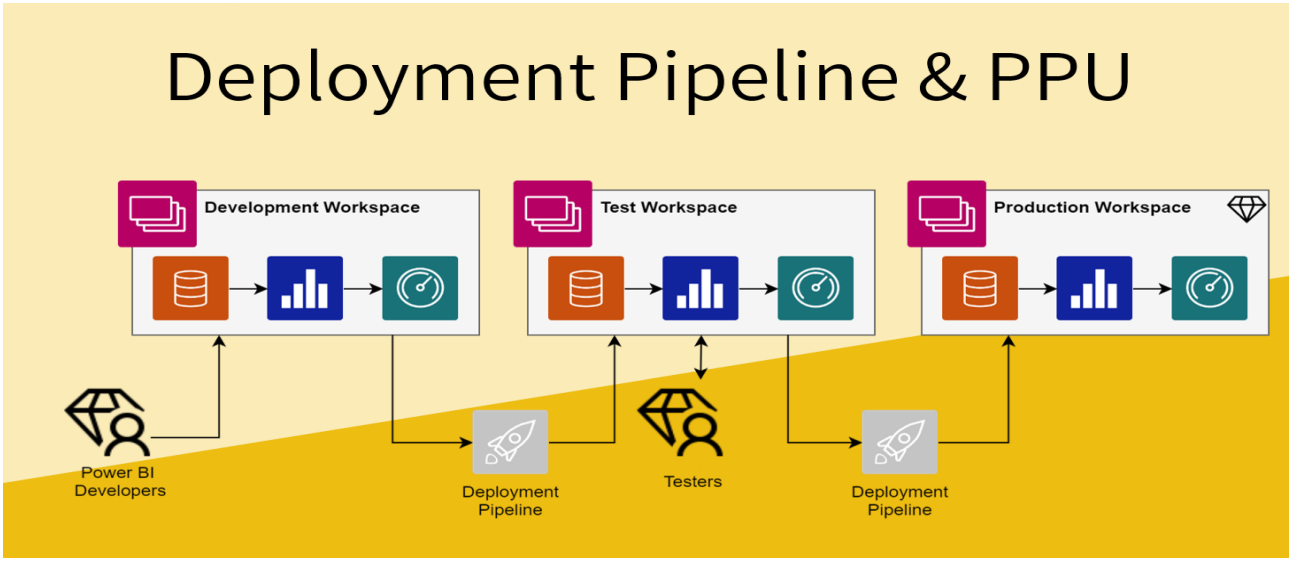
Microsoft Fabric Life Cycle Management est un ensemble d'outils qui permet aux équipes de développement de créer et de gérer des pipelines de déploiement continus (CI/CD) pour leurs applications et leurs données. Cela leur permet de livrer des mises à jour rapidement et de manière fiable, en automatisant les processus de test, de validation et de déploiement.

Fabric Life Cycle Management comprend deux composants principaux :

- L'intégration Git permet aux développeurs de connecter leurs dépôts Git à Fabric et de pousser leurs modifications vers leurs espaces de travail Fabric en continu.
- Les pipelines de déploiement automatisent le processus de déploiement des modifications apportées aux applications et aux données Fabric vers des environnements de test et de production.

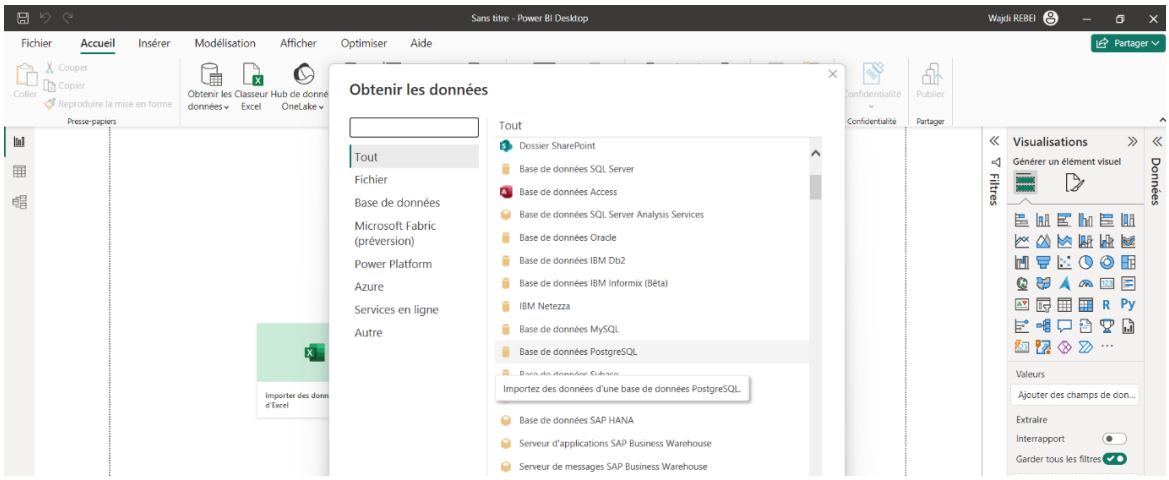
NB : Dans notre cas nous allons seulement utiliser les fonctionnalités de Repos dans Azure DevOps, nous allons donc charger tous les fichiers et tout dans Repos.

Pour le Déploiement Pipeline nous allons utiliser la fonctionnalité offerte par Fabric et ce Diagram dans la photo ci-dessous le décrit ;

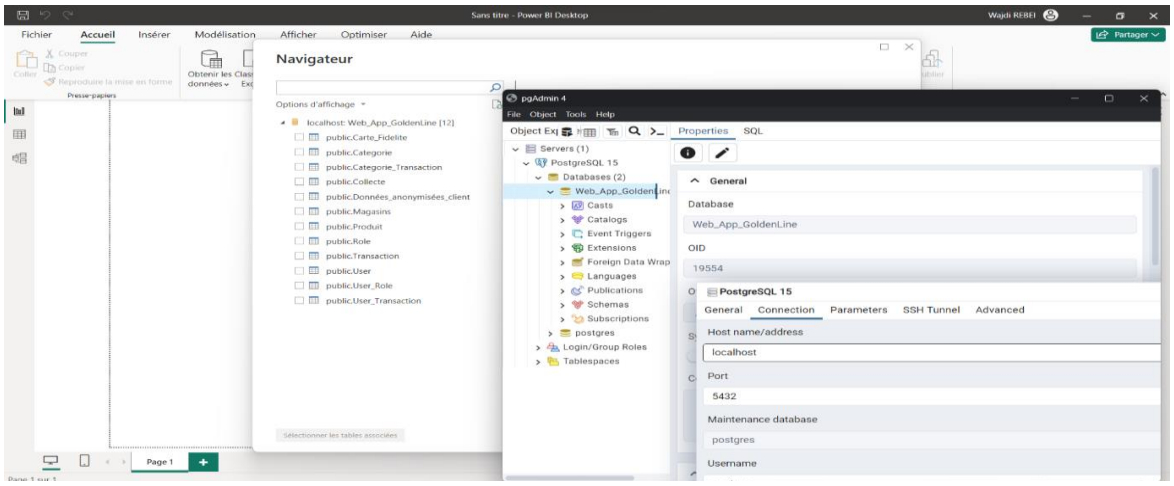


Dans cette section nous allons décrire les étapes à faire sous forme des captures d'écran aussi notre travail sera déployer sur Git après :

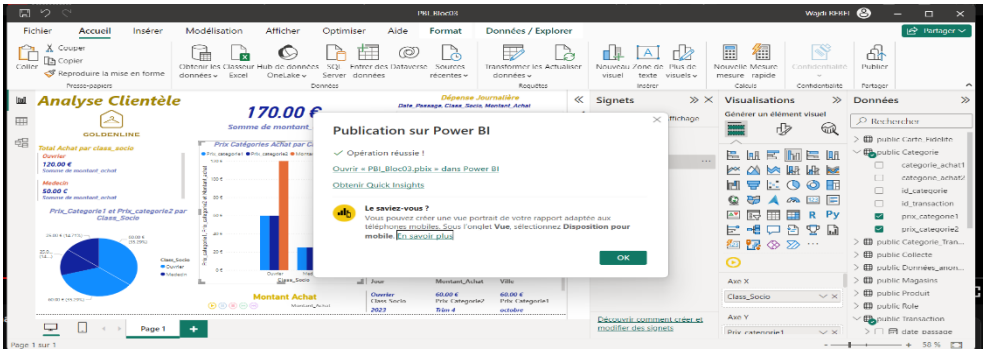
Après qu'on a installé Power BI desktop nous commençons par **Importer les données de BDD PostgreSQL** qu'on a déjà préparées auparavant comme décrite notre architecture Logicielle, normalement l'importation devra être faite en connectant Power Bi avec Azure SQL datawarehouse mais vu que on n'a pas l'accès Open Source à ceci alors nous choisissons la méthode localhost.



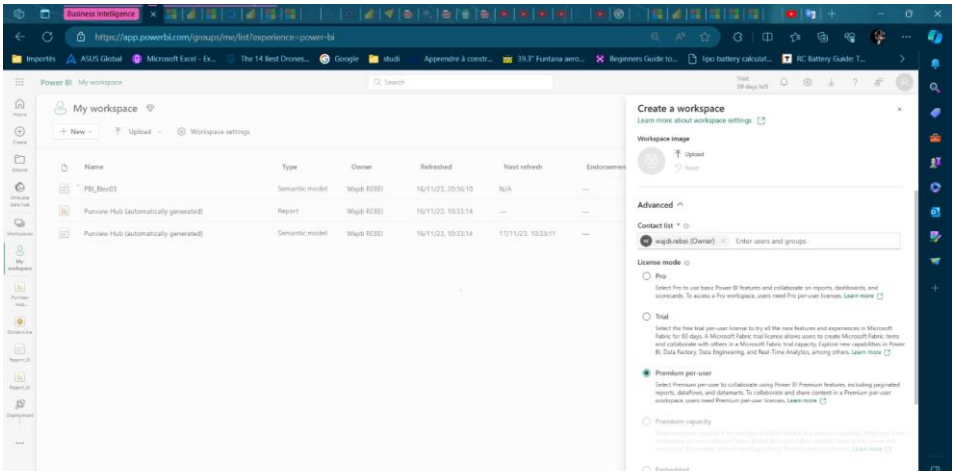
Puis nous devons configurer l'accès à BDD par le biais **pgAdmin4**



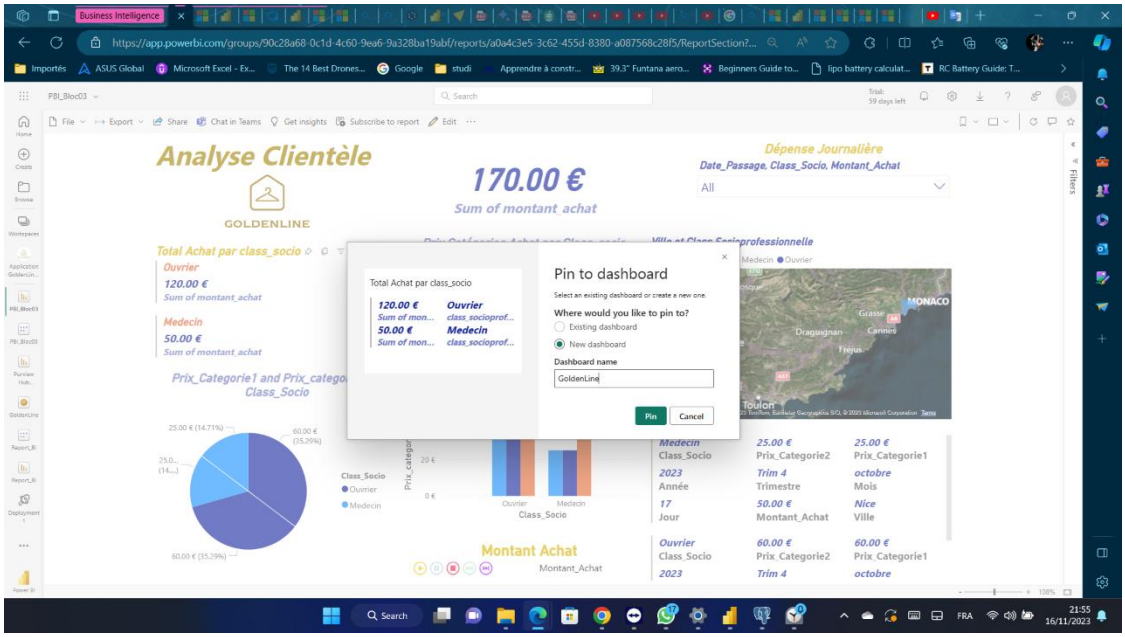
Dès que nous avons terminé la création de notre rapport dans PBI nous l'avons publié dans espace de travail PBI service. Pour rappeler ; si on veut partager notre travaux avec les utilisateurs finaux nous avons devoir créer ce qu'on appelle Application par le biais PBI service, cette application est le canal de distribution de rapports et tableaux de bord aux utilisateurs finaux. ce partage normalement doit passer par PBI service et non pas par le fichier (.pbix) sauf si les deux utilisateurs sont deux développeurs.



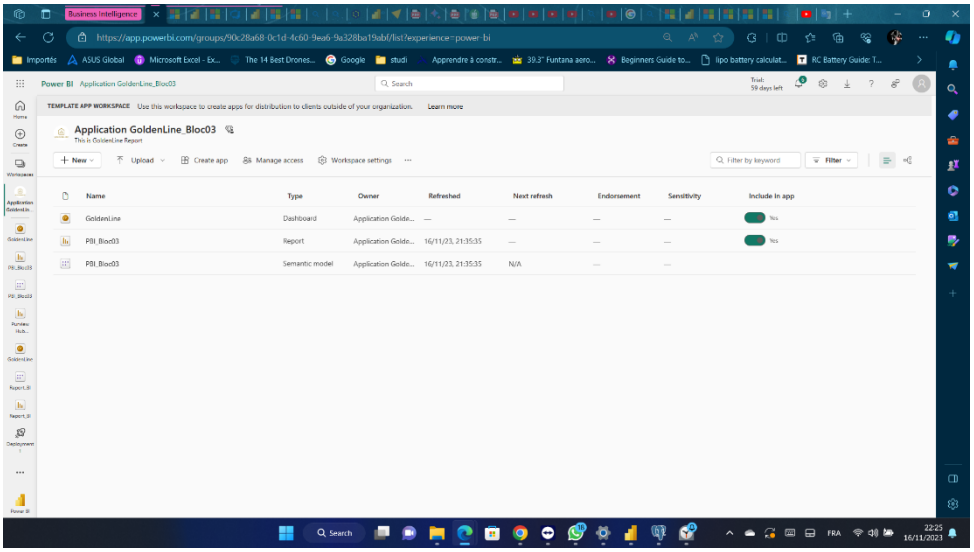
Alors nous devons enregistrer les autres utilisateurs dans un même **Workspace**. Vu que notre Power BI trial donc cette fonctionnalité nécessite une acceptation de l'administrateur et enregistrement de **Domains** DNS qui est un peu cher.



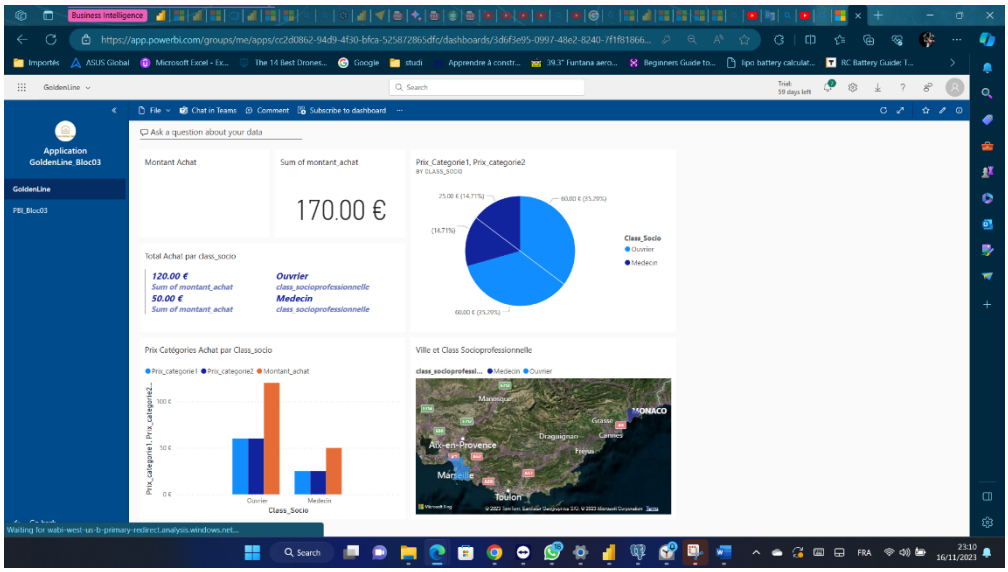
Maintenant que notre rapport est téléchargé dans PBI service on peut voir dedans le Dataset, le rapport et le tableau de bord qu'on va le créer aussi.



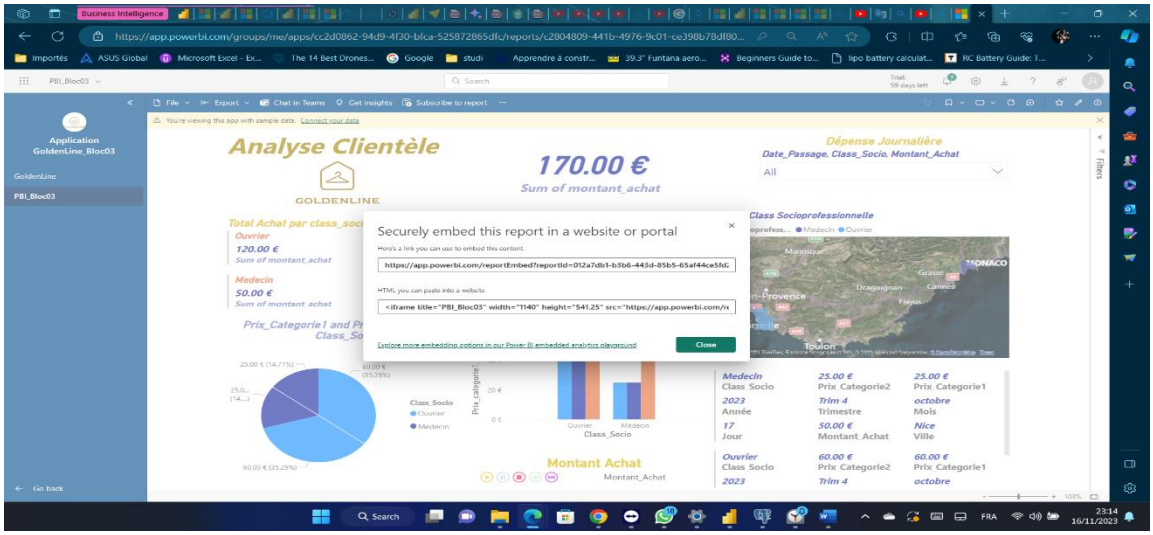
Comme indique la photo ci-dessous notre workspace 'Application GoldenLine_Bloc03' est prête et on peut créer notre application, il suffit de cliquer sur « Create app »



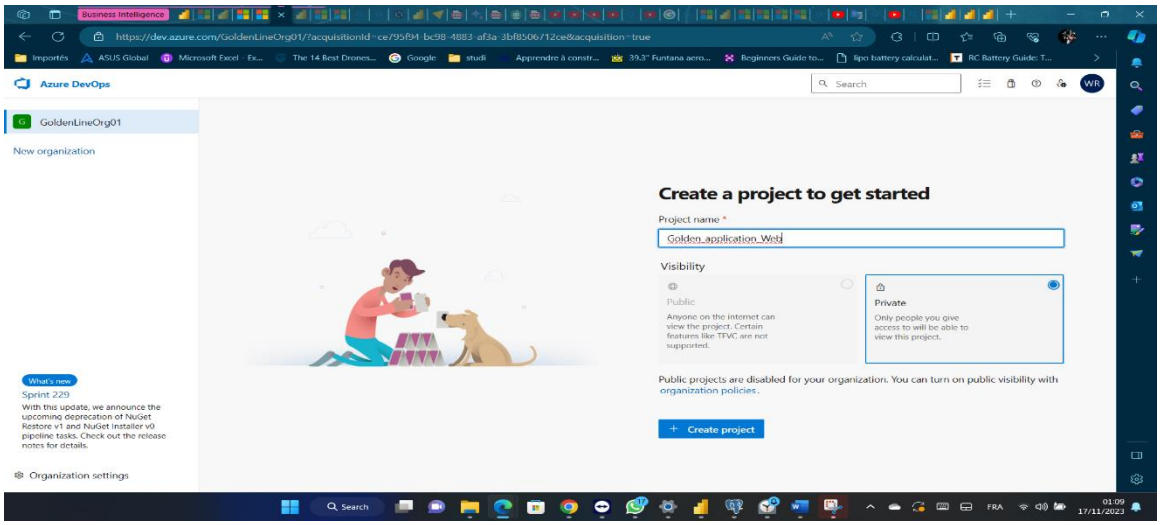
Voilà la version préliminaire de notre application web



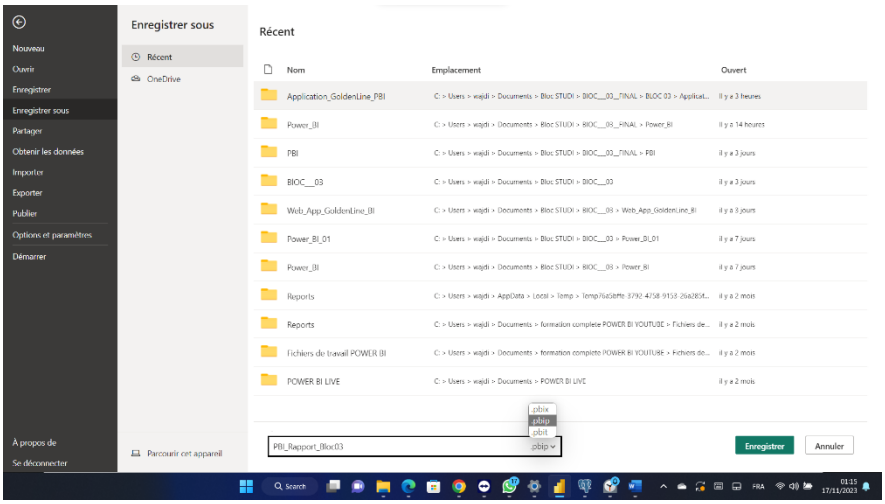
Pour partager lien de notre rapport final plusieurs options offertes par PBI service, notre intéressante c'est **Publish to web** c'est-à-dire ; on peut ensuite utiliser le lien et découvrir notre rapport ou bien copier le lien pour l'envoyer dans un e-mail, aussi copier le code HTML à coller et l'intégrer dans un site web.



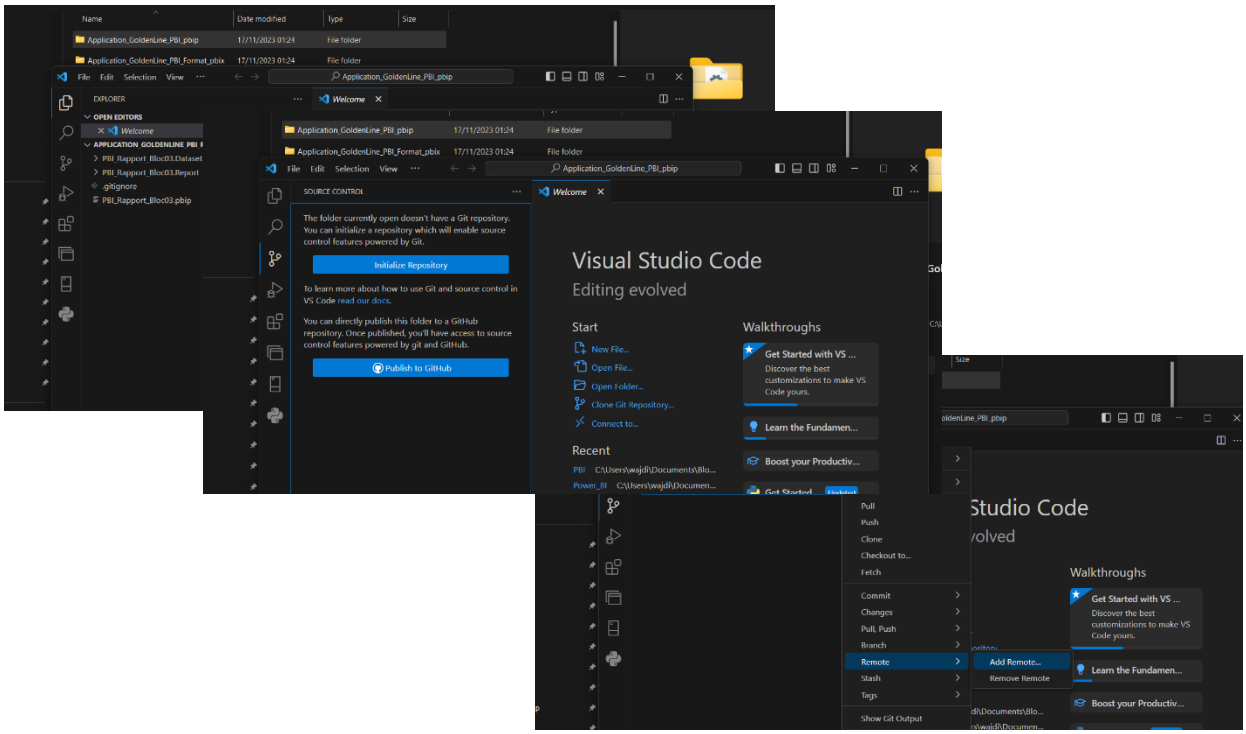
Venant maintenant à la phase de configuration d'Azure DevOps, nous allons donc créer une organisation 'GoldenLineOrg01' et notre projet 'Golden_Application_Web'

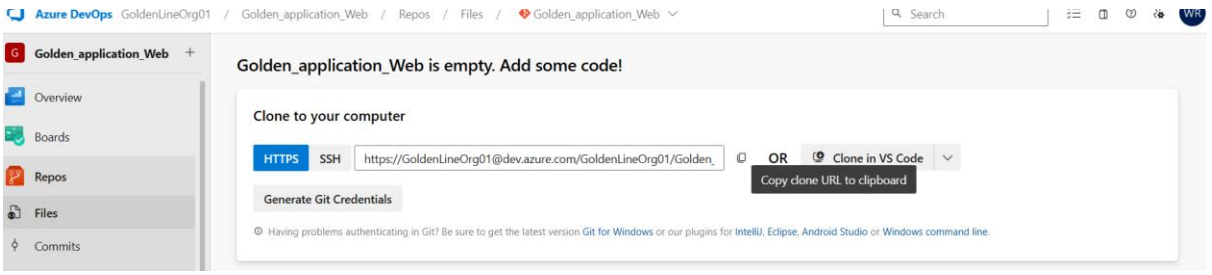


Puis on doit faire le setting repository avec VS code donc dans ce cas il faut enregistrer notre fichier rapport créer par PBI desktop sous format (.pbip) et non (.pbix)

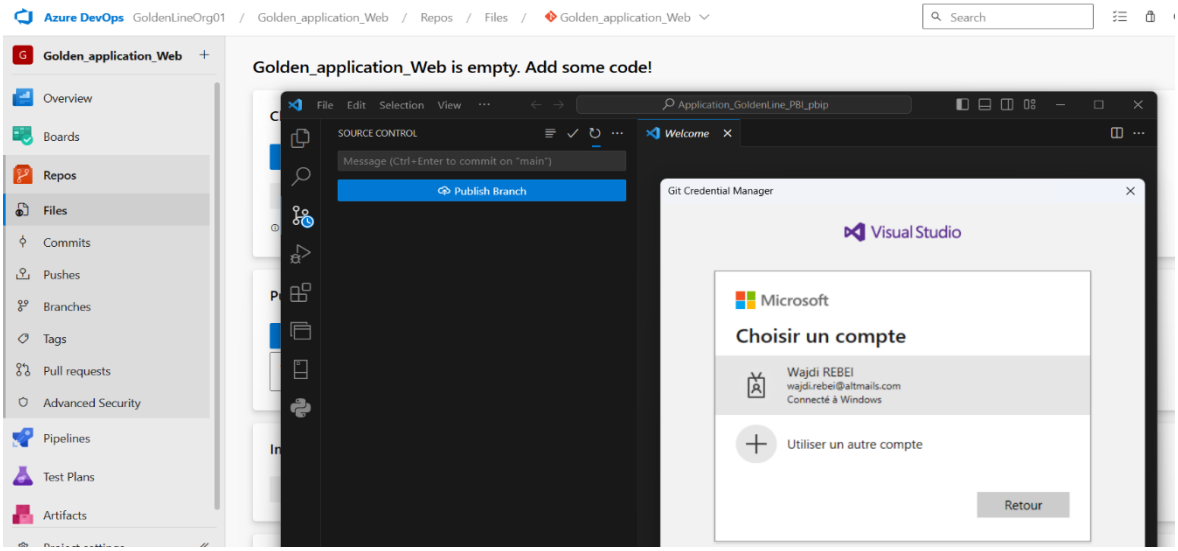


Puis suivre les étape suivant : initialize Repository - add remote - copy url to add on remote

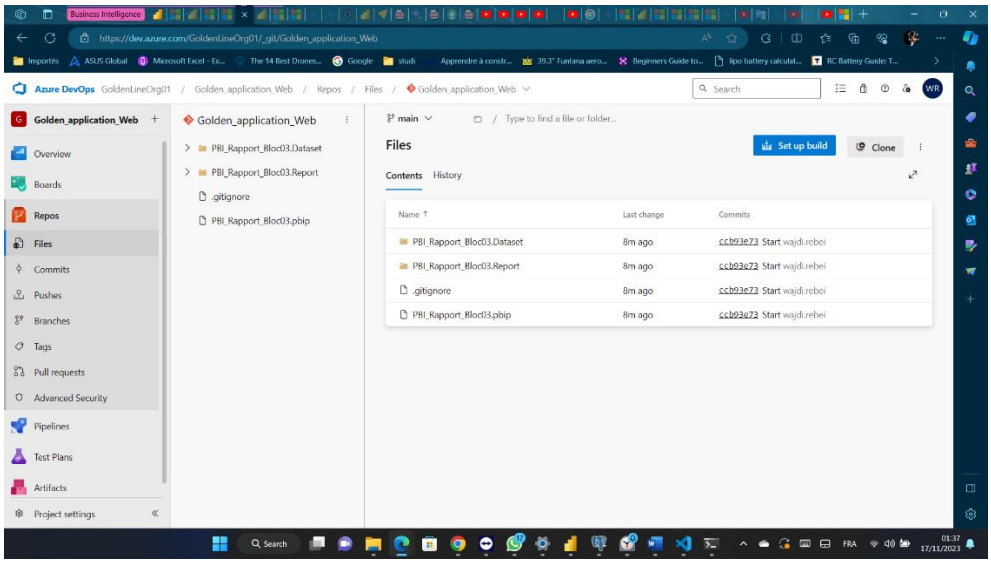




Dès qu'il sera tout prêt on exécute le 'publish to bransh'

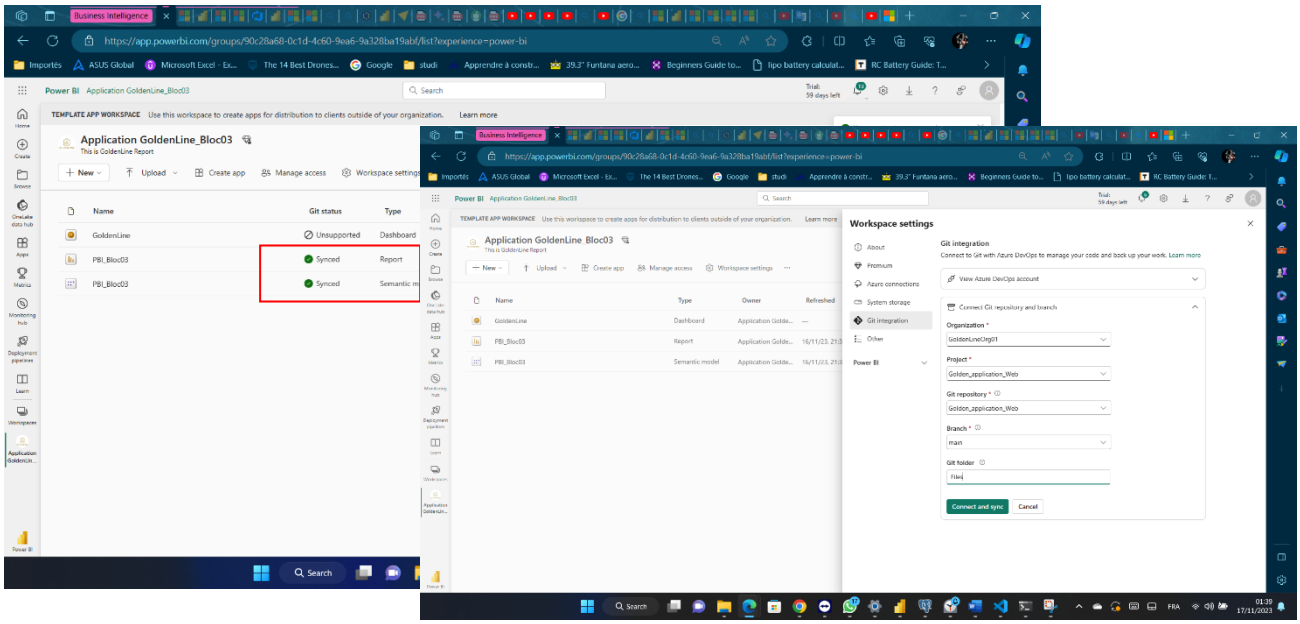


Actualisant la page Azure DevOps Repos files



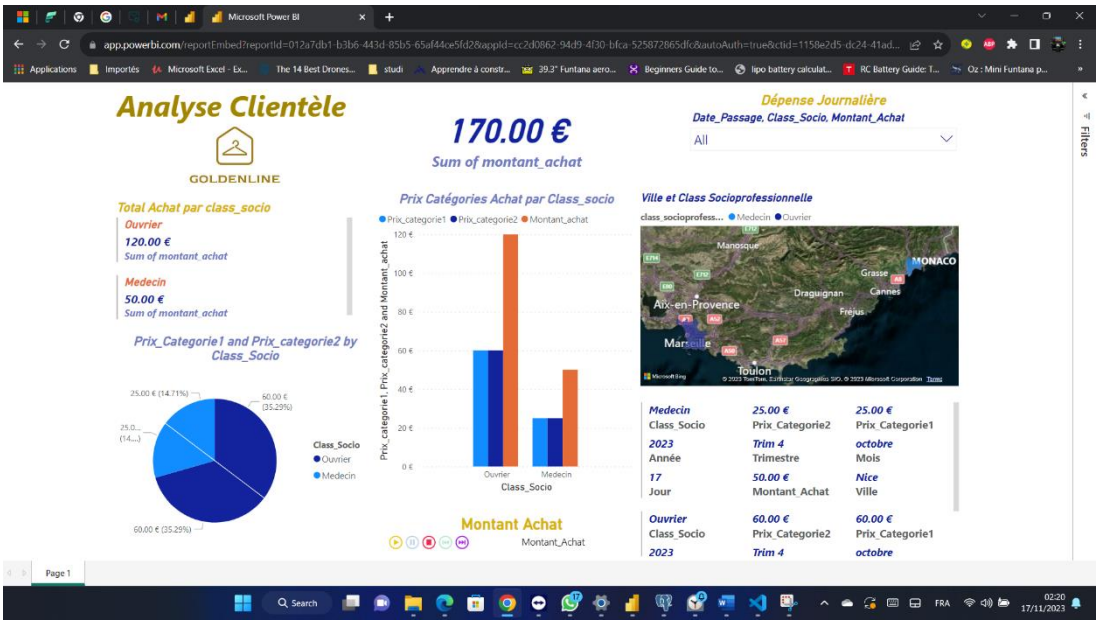
Maintenant il reste l'étape finale d'éditer les paramétrages **Workspace setting** pour synchronise **workspace PBI service** et Azure Devops organization.

Synchronization contenent 'Application GoldenLine PBI' avec 'Azure DevOps'



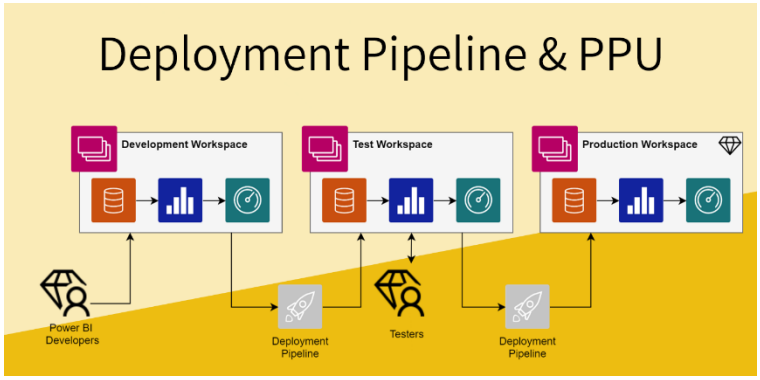
La version finale de notre Application web est comme suit (la photo ci-dessous) mais comme expliqué auparavant vu que nous n'avons pas la version payante ou l'enregistrement de domaine on ne peut pas ajouter des membres dans notre espace de travail pour partage et ou' on peut identifier leurs rôles :

- Admin
- Viewer
- Contributor
- Member

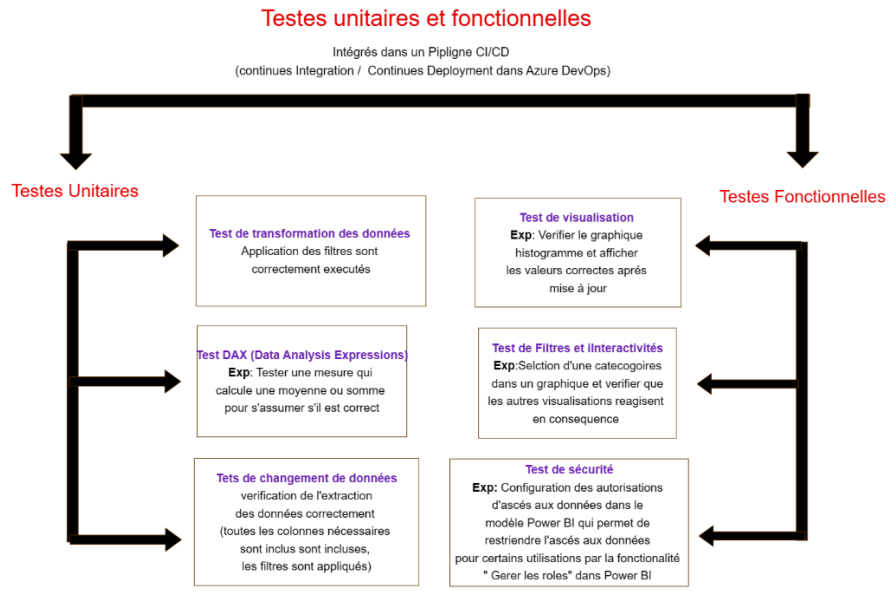


Principes déploiement pipelines dans Microsoft Fabric avec PBI service

Le diagram ci-dessous décrit un exemple de test qu'on peut le faire dans notre Pipeline CI/CD ;



Parmi les tests qu’on doit l’exécuter par exemple montrés dans la photo ci-dessous :



Le développement, les tests et la production sont trois phases essentielles du cycle de vie d'une application. Dans Fabric, ces trois phases sont bien distinctes et permettent de garantir la qualité et la stabilité de l'application.

Développement (Dev)

La phase de développement est la phase où les développeurs créent et modifient le code de l'application. Cette phase se déroule dans un environnement isolé, afin que les modifications des développeurs ne perturbent pas les autres environnements.

Tests (Test)

La phase de tests est la phase où l'application est testée pour s'assurer qu'elle fonctionne correctement. Cette phase se déroule dans un environnement de test, qui est un environnement similaire à l'environnement de production, mais qui n'est pas utilisé par les utilisateurs réels.

Production (Prod)

La phase de production est la phase où l'application est déployée et accessible aux utilisateurs réels. Cette phase se déroule dans l'environnement de production, qui est l'environnement réel où l'application sera utilisée.

Fabric permet de gérer efficacement les trois phases du cycle de vie d'une application. Il fournit des outils et des fonctionnalités qui permettent aux développeurs, aux testeurs et aux administrateurs de travailler efficacement et en toute sécurité.