BORALEX

Au-delà

DES ÉNERGIES RENOUVELABLES

Présentation des applications de l'équipe des marchés de l'énergie et des besoins DevOps

Auteur : olivier.imbaud@boralex.com
A l'attention de : Jérémie Foricheur (consultant DevOps senior)
Date de création : 13/09/2022
Date de dernière modification : 19/09/2022

Table des matières

I. Présentation de l'équipe des marchés de l'énergie

L'équipe des marchés de l'énergie, créée en 2019, a pour mission principale de répondre à un besoin stratégique de l'entreprise à savoir : la commercialisation au meilleur prix de sa production d'électricité.

Pour comprendre ce besoin, il est important de faire un zoom en arrière sur le secteur des énergies renouvelables en France. En effet, jusqu'à il y a encore quelques années, le schéma commercial de production d'énergie verte était facilement définissable. Lorsqu'un actif de production d'énergie renouvelable était mis en service, un contrat d'obligation d'achat signé par EDF entrait en application pour une durée de 15 ans pour l'éolien et de 20 ans pour le photovoltaïque. Par ce contrat, Boralex vendait l'électricité produite à prix fixe à EDF qui s'occupait de la distribuer sur le réseau géré par RTE.

Cependant, l'essor des énergies renouvelables ayant commencé en France au début des années 2000, les premiers contrats d'obligation d'achats arrivent à échéance ce qui impose de trouver de nouveaux débouchés pour valoriser la production électrique.

En conséquence, de nouveaux moyens au sein d'une nouvelle activité (le service des marchés de l'énergie) sont déployés pour la commercialisation de l'électricité.

L'équipe des marchés de l'énergie s'occupe de développer plusieurs types de commercialisation :

- A **long terme** avec les "Corporate PPA" qui sont des contrats de fourniture de gré à gré avec un consommateur ou un intermédiaire sur une durée de 1 à 20 ans.
- A **moyen terme**, c'est à dire la commercialisation de l'électricité pour une durée d'une semaine à 3 ans.
- A **court terme** en valorisant l'électricité sur les marchés physiques ce qui consiste en de la vente le jour pour le lendemain "day-ahead" ou dans l'heure pour la suivante "intra-day".

Pour apporter un support à ses activités, l'équipe des marchés de l'énergie a développé en interne trois applications :

- Une **application de gestion de portefeuilles** servant de support dans le cadre de la vente à long terme.
- **HEX** pour la vente à moyen terme.
- **VPP** pour la vente court terme.

II. Informations pratiques

a. Application VPP

Le contact de référence est : *simon.gesret@boralex.com*

La documentation complète de l'application VPP est disponible sur le SharePoint de l'équipe des marchés de l'énergie au lien suivant : Documentation Application VPP. (Si besoin, demander les accès du SharePoint à un des membres de l'équipe.)

De façon générale, toutes les informations relatives à la VPP (procédures de déploiement, logins des VM, BDD, Docker, GitHub …) sont accessibles sur le SharePoint de l'équipe au chemin ci-après :
Documents > Equipe Marchés > 02_Agrégation > 02_VPP-Application_et_Cadre_Technique

Les repositories GitHub de l'application VPP sont disponibles aux liens suivants :

- Backend : https://github.com/Boralex-France/blx-vpp_backend
- Frontend : https://github.com/Boralex-France/blx-vpp_frontend
- CI avec Jenkins (non fonctionnel) : https://github.com/Boralex-France/blx-vpp_ci

L'application (Backend et Frontend) est hébergée sur les serveurs suivants :

- Production : 10.140.242.102
- Test : 10.140.242.82

Les bases de données MySQL sont localisées sur les serveurs suivants :

- Production : 10.140.242.103
- Test : 10.140.242.101

Le Dashboard est visible aux adresses suivantes :

- Production : http://10.140.242.102:8000/
- Test : http://10.140.242.82:8000/

Le monitoring des flux du backend de l'application est consultable aux adresses suivantes :

- Production : http://10.140.242.102:8080/
- Test : http://10.140.242.82:8080/

Concernant les permissions, les logins de l'application et des infrastructures, je préfère te laisser voir ça directement avec Simon.

b. Application HEX

Le contact de référence est : *hind.elwafi@boralex.com*

Une présentation Power Point de l'application est disponible sur le SharePoint de l'équipe des marchés de l'énergie au lien suivant : Documentation_HEX.

Le repository de l'application est sur Google Cloud Plateform (voir avec Hind pour plus d'informations)

L'application est à la fois hébergée sur le Cloud et en local.

En local, l'application est localisée sur le serveur :

- 10.140.242.86

Le Dashboard est visible à l'adresse suivantes :

- boralex2020.appspot.com

Concernant les permissions, logins de l'application et des infrastructures, je préfère te laisser voir ça directement avec Hind.

c. Application gestion de portefeuilles

Les contacts de référence sont : *hermann.ngayap@boralex.com* et *youcef.khelif@boralex.com*

Le repository GitHub de l'application de gestion de portefeuilles est disponible au lien suivant :

- Backend et Frontend : https://github.com/Boralex-France/PPA_Dashboard

L'application est localisée sur le serveur (à date, au moins pour ce qui est du Frontend) :

- 10.140.242.86

La base de données MySQL Server est hébergée sur le serveur :

- 10.140.251.3

Le Dashboard est visible à l'adresse suivante :

- http://10.140.242.86:8070/

Concernant les permissions, logins de l'application et des infrastructures, je préfère te laisser voir ça directement avec Joël, Hind, ou bien Simon.

d. Logiciels utiles à installer

Les outils ci-dessous ne sont pas tous indispensables, libre à toi de te constituer l'environnement de travail qui te convient le mieux :

- WSL Ubuntu pour un accès en local au système Linux
  Docker Desktop for WSL pour faire le lien entre docker et le système linux
- VSCode de mon point de vue le meilleur IDE (possibilité de connexion à Docker, GitHub, BDD …)
- GitHub Desktop pour faire le dialogue entre les repositories distants et tes repositories en local (personnellement j'utilise git directement sur le shell)
- nRemoteNG pour se connecter aux différentes VMs hébergeant les applications et les bases de données
- Dbeaver pour pouvoir visualiser facilement les tables et les données des différentes bases
- Filezilla pour transférer des fichiers depuis l'environnement local vers les VMs

Remarque : Il est possible que tu ne puisses pas installer toi-même ces outils sur ta machine, il faudra probablement voir ça avec les TI : helpdesk@boralex.com .

e. Support IT

Pour des problématiques plutôt d'infrastructures / sécurité, les personnes à contacter sont : Emmanuel Leu et Mario Lessard.

Pour les problème informatiques plus conventionnels, il faut faire des tickets en passant par l'application ti.Ger ou via le mail indiqué dans la partie précédente.

III. Présentation de l'application VPP

a. Architecture générale

L'application VPP est constituée de trois composants fonctionnant en parallèle :

- Le Backend dont le rôle est de récupérer, formater et stocker automatiquement les données utilisées dans le cadre de la vente d'électricité à court terme.

- Le Frontend qui correspond à un dashboard consultable par des utilisateurs est disponible en ligne.

- La base de données MySQL permettant de stocker l'intégralité des informations formatées : prix, production, prévisions, ordres, trades, données d'entrées des modèles de prévision interne et autres métadonnées.

La figure n°1 ci-après présente l'architecture générale de l'application VPP :

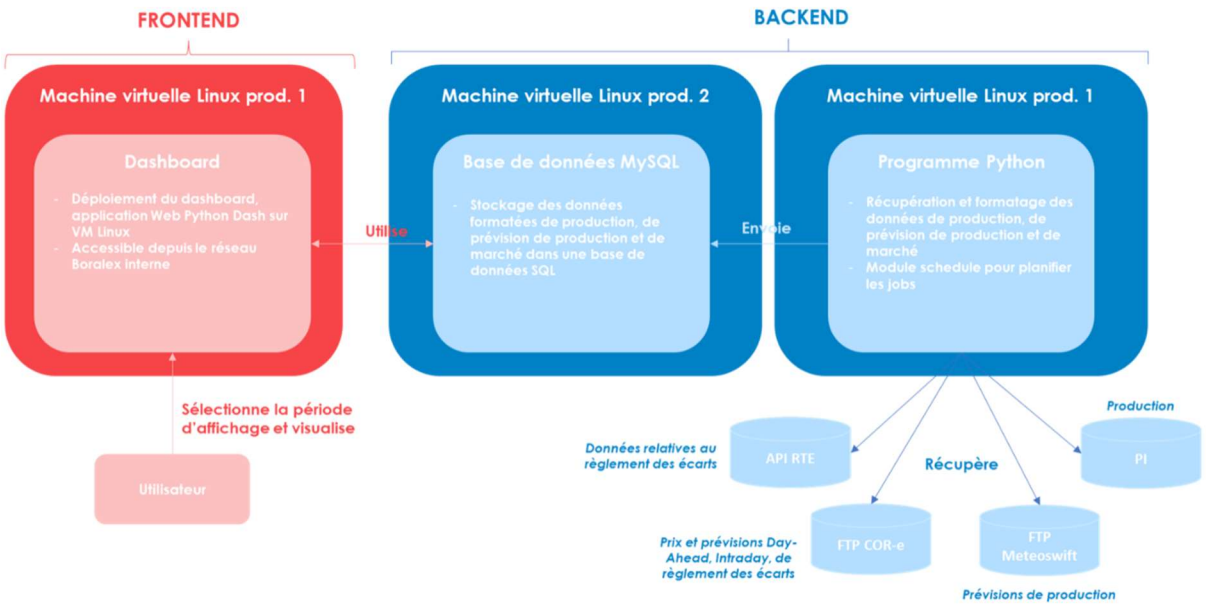

Figure n°1 : Architecture de l'application VPP. Seuls quelques flux de données sont représentés.

L'ensemble du back-end a été développé sous Python 3.8.10 et est containerisé, via Docker et un environnement virtuel Anaconda. L'automatisation des tâches est assurée par l'orchestrateur Prefect dans sa version Core, gratuite et hébergée localement. Prefect met à disposition un dashboard pour contrôler l'exécution des flux de l'application. Le versionnage du code est réalisé via GitHub avec un stockage sécurisé dans le Cloud.

Les images Docker générées pour le back-end sont mises à disposition des différents environnements par l'intermédiaire d'un espace Docker Hub, lui aussi dans le Cloud. Les machines virtuelles hébergeant les processus applicatifs sont fournies par Boralex, et de type Linux (Debian 11).

L'intégralité des données transite et/ou sont stockées dans la base de données MySQL, hébergée dans une machine virtuelle dissociée de celle contenant l'application. Des requêtes SQL sont donc intégrées dans des algorithmes Python afin de mettre en ligne et de récupérer les données formatées.

Le front-end est lui aussi développé en Python 3.8.10 via le module Dash (framework de développement d'applications Web basé sur Flask, React.js et Node.js) dans un environnement virtuel miniconda spécifique et facilement exportable. Des feuilles de style CSS permettent de personnaliser l'aspect des différents éléments. De la même façon que pour le back-end, le versionnage du code est réalisé avec Github et un container Docker dédié au front-end a été mis en place.

En ce qui concerne les données externes, Le tableau n°1 ci-après présente les principales sources :

Sources	Techniques	Types de données
RTE	API	Prévisions de consommation d'électricité
		Prévisions de production d'électricité
		Prévisions de d'indisponibilités du réseau
		Règlement des écarts
ELIA	API	Prévisions de production d'électricité
ENTSOE	API	Prévisions de consommation d'électricité
		Prévisions de production d'électricité
date.nager.at	API	Jours fériés
EZnergy	API	Récupération des ordres intra-day (environnement de test et de production)
		Envoie des ordres intra-day (environnement de test et de production)
NordPool	API	Récupération et envoie des ordres day-ahead
		Récupération de la facturation
COR-e	FTP	Prix day-ahead, intra-day, règlement des écarts
		Prévisions day-ahead, intra-day, règlement des écarts
Météoswift	FTP	Prévisions de production
www.powernext.com api.investing.com	Web scraping	Prix du gaz et coût du carbone

Tableau n°1 : Sources et types des données externes utilisées par l'application VPP.

b. Procédure de développement de l'application VPP

Le développement du back-end et du front-end repose sur le logiciel GitHub qui permet de versionner le code de l'application dans le Cloud et de collaborer entre développeurs. La procédure de développement repose sur une stratégie de branching standard et conforme aux bonnes pratiques, présentée en figure n°2 ci-après.

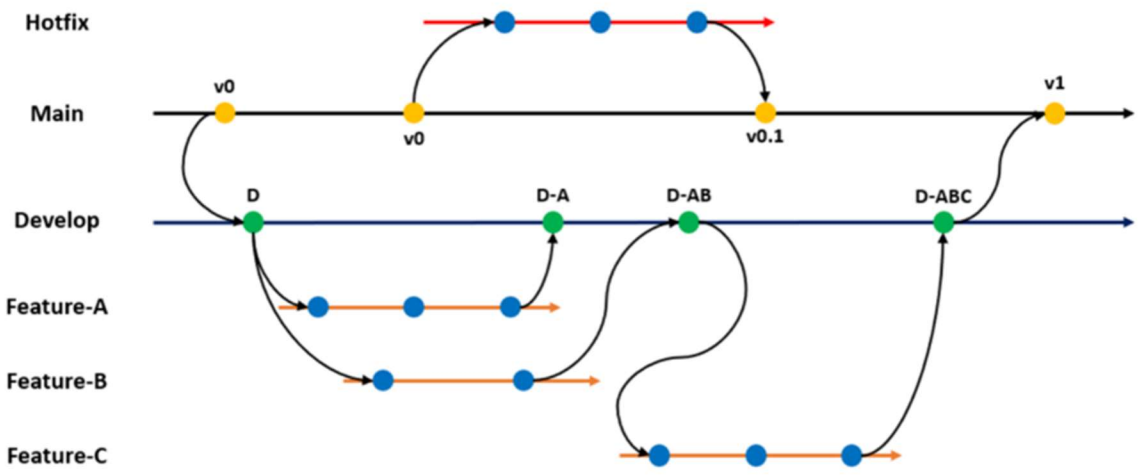

Figure n°2 : Stratégie de branching GitHub pour le développement de l'application VPP.

Description des rôles des différentes branches :

- **Master** : Cette branche, aussi parfois appelée Main, est la branche la plus importante de l'application. En dehors de cas exceptionnels, personne ne doit développer directement dessus. Elle représente la version de production de l'application et doit donc toujours être fonctionnelle.
- **Develop** : Cette branche représente la version de test de l'application et elle est créée à partir de Master. C'est à partir de là que sont développées les nouvelles fonctionnalités de l'application avec les branches Features.
- **Feature** : Cette ou ces branches sont créées à partir de Develop et visent à ajouter une nouvelle fonctionnalité à l'application. Leur nomenclature doit être explicite, concise et uniforme.
- **Hotfix** : Cette branche est exceptionnellement créée à partir de Master lorsqu'un bug critique est détecté et doit être corrigé très rapidement. Il doit être bref et ne contenir que des commits liés au bug. Cet évènement doit conduire à repenser des procédures de test plus drastiques au niveau de la branche Develop.

La méthode de développement vise à respecter l'ordre hiérarchique des branches et d'informer les autres développeurs lors de la fusion d'une branche Feature vers la branche Develop et surtout, de le branche Develop vers la branche Master.

Pour cela les développeurs de l'application utilisent des pull-request qui permettent d'inviter d'autres développeurs à valider les modifications avant que la fusion ne soit effective. La pull-request offre un véritable espace convivial où chacun peut donner son avis et proposer des correctifs ou des améliorations, la communication étant la clé pour éviter bien des problèmes. Une fois la fonctionnalité validés et fusionnée, les branches Feature inutilisées sont supprimées.

Remarque : Pour tester les fonctionnalités implémentées sur une branche GitHub donnée, il est nécessaire de se placer en local dans le conteneur Docker de l'agent. Ce dernier est construit à partir de l'image Docker de l'application faisant référence à ladite branche.

c. Infrastructures

Le développement de l'application est réalisé en local sur la machine du développeur. L'application fonctionne sur quatre machines virtuelles (VMs) Linux différentes : deux pour les tests, et deux pour la production. Sur chaque environnement, une VM est dédiée à la base de données, et une autre à la partie applicative. Ces VMs sont hébergées en France sur des serveurs situés à Blendecques et sont gérées par le personnel IT de Boralex. A noter que pour des raisons de sécurité, les VM sont dupliquées sur des serveurs de Boralex au Canada, il y a donc techniquement au total 8 VMs et serveurs. La figure n°3 ci-après montre l'environnement de test et de production :

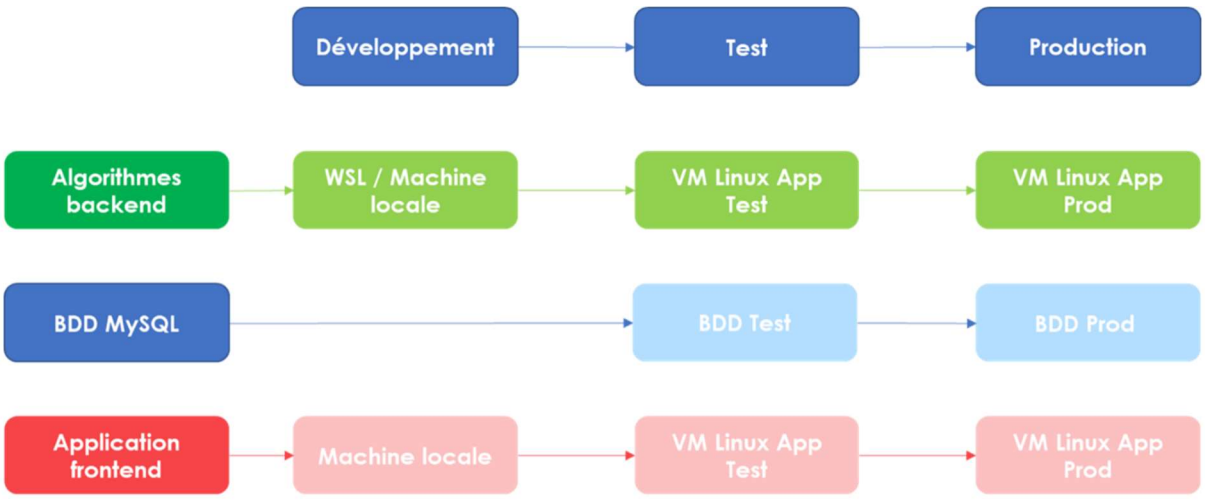

Figure n°3 : Environnement de test et de production.

d. Coopération avec les services liés

L'équipe de développement de VPP interagit avec le personnel IT (Emmanuel Leu et Mario Lessard) pour la gestion des machines virtuelles et les problématiques liées à la sécurité informatique. Un groupe Teams a été créé spécialement à cet effet pour faciliter la coopération entre les deux services.
L'équipe est aussi en contact avec le service de comptabilité pour la facturation correspondant à l'activité de vente de l'application VPP. Le rapport de facturation est généré automatiquement chaque jour par l'application et est transmis par mail au service de comptabilité avant 16h afin d'éviter toute intervention manuelle.

IV. Présentation de l'application HEX

Voir le lien vers le Power Point dans la section « Informations pratiques » ou directement avec Hind.

V. Présentation de l'application de gestion de portefeuilles

a. Description de l'architecture

L'application de gestion de portefeuille est constituée de 4 principaux composants :

1. Des ETL développés avec Python qui récupèrent les données de production, de prix de contrats (OA, CR et PPA), les données des projets en planification, la cotation journalière des produits futurs cal, trimestriels et mensuels.

2. Des ETL Pentaho qui récupèrent les données transformées et structurées obtenues en 1 et les chargent dans la base de données. **NB : une version python/SQL est en développement pour remplacer les étapes réalisées sur Pentaho car ce dernier est difficile à mettre en place sur une VMs Linux.**

3. Une base de données MS SQL SERVER pour stocker les données structurées.

La base de données est composée de 10 tables :

- **Asset** : contient les données des parcs en exploitation et ceux en planification
- **Hedge**: contient les données des contrats de couverture
- **p50_p90_asset** : contient les données des volumes de production mensuels des parcs en exploitation et ceux en planification.
- **p50_p90_hedge** : contient les données des volumes des contrats de couverture
- **contract_prices** : contient les données des prix des contrat de couverture
- **market_prices** : contient les données des prix de marché
- **market_prices_fr_eex** : Contient les données de prix des produits mensuels obtenu à travers la courbe d'extrapolation des prix.
- **mark_to_market** : Contient les données de mark to market historique. Notamment la somme des mtm annuels du portefeuille merchant. Calculer chaque jour suivant la formule volume hedge*(prix contrat-prix marché).
- **dates_calender**: Contient les différentes dates de jour de l'horizon temps.
- **datetime** : Contient les données de dates mensuelles de l'horizon temps.

Le tableau ci-dessous présente une description des différentes tables de la base de données :

Tables	colonnes
asset	projet_id: identifiant unique pour chaque actif asset_id: identifiant unique de chaque actif projet: Nom du parc technologie: éolien ou solaire Cod: date de mise en service du parc mw:puissance installée non pondérée taux_succès: probabilté que le parc entre en exploitation puissance_installée: mw*taux_succès eoh: Expecting Opearting Hour (nombre d'heure en operation/an) Date_merchant: date de sortie de contrat OA ou CR Date_dementelemnt: date de démantèlement du parc Date_msi:date de mise en service P50: volume annuel de production (prob à 50%)

	P90:volume annuel de production (prob à 90%)
hedge	hedge_id: identifiant unique de chaque contrat de couverture Projet_id: identifiant unique de chaque parc Type_contrat: type de contrat de couverture OA, CR ou PPA Date_debut: Date de debut du contrat Date_fin: date de fin du contrat Profil: Pct_couverture: Pourcentage de la production vendu Contrepartie: acheteur de la production Pays_contrepartie: Pays d'origne de l'acheteur
Contracts_prices	Hedge_id: identifiant unique de chaque parc Projet_id: identifiant unique de chaque Date: date de notre horizon temps (fréquence mensuelle) Price:prix de vente (contrat OA, CR, PPA)
P50_p90_asset	Asset_id: identifiant unique chaque actif Projet_id: identifiant unique chaque parc Date: date p50: production mensuelle (p50 annuel*profil mensuel de production) p90: production mensuelle (p90 annuel*profil mensuel de production)
P50_p90_hedge	hedge_id: identifiant unique de chaque contrat de couverture Projet_id: identifiant unique chaque parc Date: date p50: volume vendu (p50 annuel*profil mensuel de production*pct_couverture) p90: volume vendu (p90 annuel*profil mensuel de production*pct_couverture)
Market_prices	Hedge_id : identifiant unique de chaque contrat de couverture Projetid : identifiant unique de chaque parc Delivery_period: date de l'horizon Settlement_price: prix de marché produit mensuel
datetime	dates_id: dates de l'horizon temps (2022-2028) fréquence mensuelle
calender	dates_id: dates de l'horizon temps (2022-2028) fréquence journalière
market_prices_fr_eex	delivery_period: Date hozizon temps settlement_price : Prix de marché produit mensuel de la courbe de prix cotation_date : Date de cotation
mark_to_market	date: date de calcul du mtm Mtm:valeur du mtm historique

4. Un Dashboard interactif Dash plotly pour la visualisation des indicateurs du portefeuille :

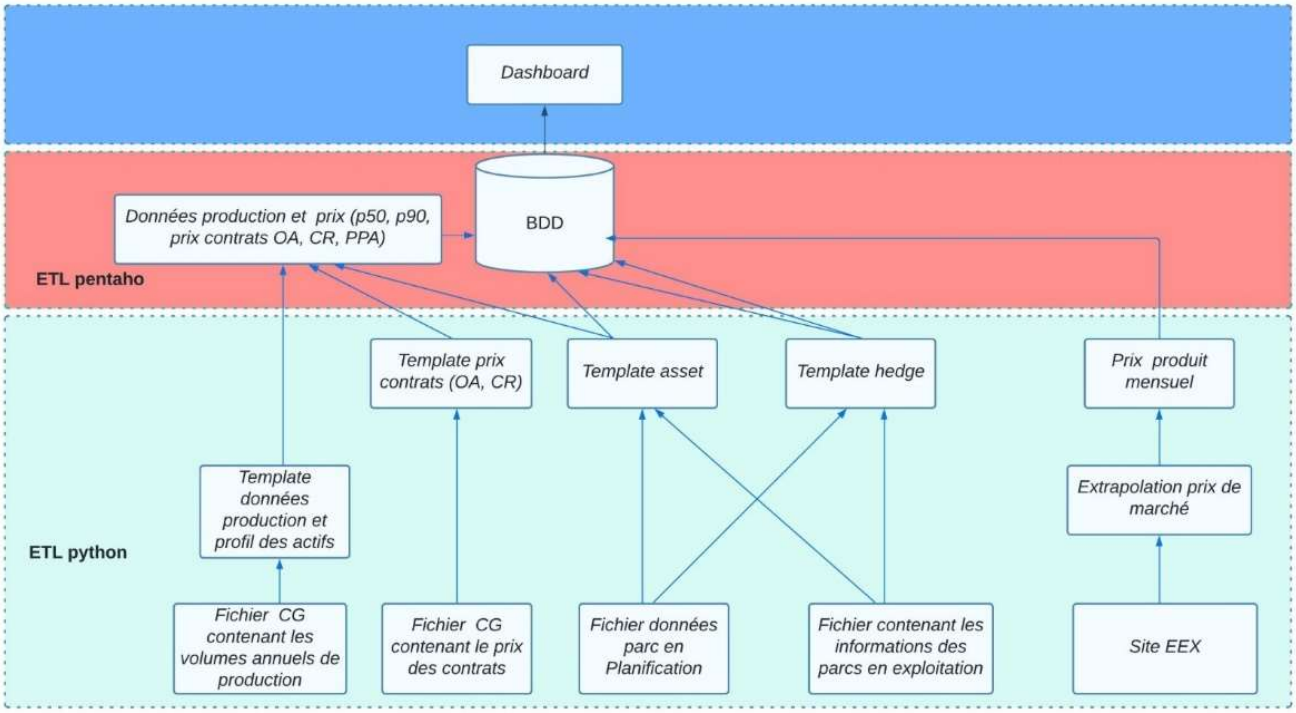

Figure n°4 : Architecture globale application de gestion de portefeuille

Les outils nécessaires au fonctionnement de l'outil sont :

- MySQL/MS SQL SERVER
- Workbench/MS SERVER Management Studio
- Pentaho Data Integration
- IDE Spyder

b. Acquisition et chargement des données

L'ensemble des codes a été développés sous python.

En début du processus, il y'a 4 principales sources de données :

- Un fichier Excel qui contient les données de production ainsi que les profils mensuels de production de chaque parc.
- Un fichier Excel qui contient les prix des contrats.
- Un fichier Excel qui contient les données des projets en développement.
- Un fichier Excel qui contient les données des parcs déjà en exploitation.

Ordre d'exécution des scripts pour la création des templates. Certains serviront d'inputs à la création d'autres :

1. Exécuter le script « pipeline_to_create_template_prod » Pour créer un premier template_prod temporaire qui contient les données de production (p50, p90 annuel), les données de profil de chaque parc ainsi que les profils typiques éolien et solaire.
2. Exécuter le script « pipeline_to_create_template_asset » afin de générer le template_asset temporaire qui ne contient pas les données de production annuelles.

De ce fichier temporaire dénommé « asset_vmr_planif » et présent dans le dossier temporaire. Les colonnes « projet_id et projet » y seront extraites et serviront d'inputs pour le scripte suivant qui rajoute les projets_id aux données de profil mensuuels.

3. Exécuter le script « pipeline_to_modify_template_prod ». Pour rajouter les projet_id aux données de production annuelles, et remplacer le nom des parcs par l'identifiant unique (projet_id) de chaque parc correspondant dans la feuille prod_perc (profiles mensuels de produdtion) du template_prod.

Il y a 4 parcs qui entrent en exploitation en cours d'année 2022 notamment (Bois des Fontaines (BDF), Repowering Bougainville (BGV-R), Repowering Evits et Josaphats (EEJ-R), Repowering Remise de Reclainville (RCL-R)) Nous devons modifier les données de production de ces parcs manuellement en remplaçant leurs p50 et P90 par respectivement 55600, 47000 ; 48100, 42300 ; 42100, 36800 ; 40800 35300.

Le fichier « template_prod » est ainsi créé et servira d'input pour le calcul des volumes mensuels.

4. Exécuter le script « pipeline_to_load_template_asset » ajouter les données p50 et p90 annuelles au template asset temporaire céée en (2) pour générer le « template_asset » definitif qui sera chargé dans la base de données pour la première fois dans la table asset.

5. Exécuter le script « pipeline_to_create_template_hedge » pour créer le template_hedge

6. Exécuter le script « pipeline_to_load_template_hedge » pour charger les données du template_hedge dans la base de données.

7. Exécuter le script « pipeline_p50_asset » pour calculer le p50 et p90 des assets et pour charger les données de p50 et p90 mensuelles pour tous les actifs dans la table p50_p90_asset de la base de données.

8. Exécuter le script « pipeline_p50_hedge » pour générer les volumes de couverture calculés en fonction du type de contrat, du pourcentage de couverture des dates de début et de fin des contrats. Ensuite charger ces données de volumes dans la base de données dans la table p50_p90_hedge.

9. Exécuter le script « pipeline_to_create_template_prices » Pour générer un template qui contient les prix des contrats (OA et CR) pour tous les parcs en exploitation.

10. Exécuter le script « pipeline_to_load_contract_prices » pour assigner à chaque actif en production un prix mensuel pour le contrat en foction du contrat de couverture, pour des parcs en planification emmettre des hypothèses de prix en fonction de la technologie du parc (eolien ou solaire). Y ajouter les prix des contrats PPA. Faire une projection sur l'horizon temps l'horizon 2022-2028. ET enfin charger ces données dans la base de données dans la table « contracts_prices ».

Description des templates :

Templates	Exemples de données
Asset	Puissance installée, date COD, date merchant, date démantèlement, taux de succès, eoh
hedge	Type de contrat, pourcentage de couverture, contrepartie, date de début, date de fin, technologie (éolien, solaire), date de démantèlement
prices	Prix mensuel
Production	P50, p90, profil mensuel, profil éolien, profil solaire

c. Fonctionnement

Les données sont stockées de manière structurée dans des table de la base de données.
Une connexion directe est établie entre la BDD et le Dashboard à travers un module python qui contient toutes les requêtes SQL. Pour chaque graphe du Dashboard, une requête spécifique a été créée. Le résultat de la requête est sous forme de data frame qui contient les données qui seront représentées sur le Dashboard.
Lorsque l'application dash est démarrée, le résultat de chaque requête est renvoyé sous forme de data frames et chargés dans la mémoire vive de spyder.

d. Mise à jour des tables

La mise à jour des tables s'effectue à travers des scripts dévéloppés sous python. La fréquence de mise à jour d'une table varie considérablement d'une à l'autre :

- La table **asset** ne nécessite pas une mise à jour fréquente. La màj peut s'effectuer seulement quand il y-a des nouveaux parcs qui proviennent delà planification et qui doivent être intégrés au portefeuille. C'est une table de dimension à changement lente de type 1. c.à.d lors du processus de màj, Les données du flux source sont insérées dans la table de destination et écrasant l'ancienne version. Pas besoin de conserver un historique des données.
- La table **hedge** en revanche doit être mise à jour assez régulièrement notamment lorsque de nouveaux contrats sont signés ou quand les informations d'un contrat sont modifiées. Cependant c'est une table de dimension changeante lente de type 2. c.à.d lors du processus de màj. L'historique de la table est conservé. Une nouvelle donnée d'un champ en provenance du flux entrant sont détectées, les données de la table sont màj et l'ancienne version est conservée et les nouvelles données sont introduites dans la BDD.
- La table **p50_p90_asset**: Cette table est une table de type 1. Lorsque de nouvelles données d'un champ en provenance du flux entrant sont détectées, les données de la table sont remplacées par celles-ci. L'ancienne version n'est pas conservée.
- La table **p50_p90_hedge**: Cette table aussi est une table de type 1. Lorsque de nouvelles données d'un champ en provenance du flux entrant sont détectées, les données de la table sont remplacées par celles-ci. L'ancienne version n'est pas conservée.
- La table **contrat prices**:
- La table **market prices**: Cette table est également une table à évolution lente de type 2. Lors du processus de màj, L'ancienne version des données présentes dans la BDD est conservée. Les nouvelles données y sont introduites

VI. Besoins DevOps

a. Besoins de l'application VPP

La mise en place d'un pipeline DevOps est la prochaine étape pour accélérer l'industrialisation de l'application VPP. Cette approche permettra de fiabiliser et de faciliter l'évolution de l'application par la construction d'un pipeline d'intégration et de déploiement continue (CI/CD) avec des outils (qui restent encore à définir et à challenger) tels que Jenkins, GitLab CI, Circle CI, Ansible, Kubernetes ... Les figures n°5 et n°6 ci-après présentent respectivement le pipeline actuel et le pipeline cible (les outils utilisés pourront varier mais le principe d'automatisation reste le même).

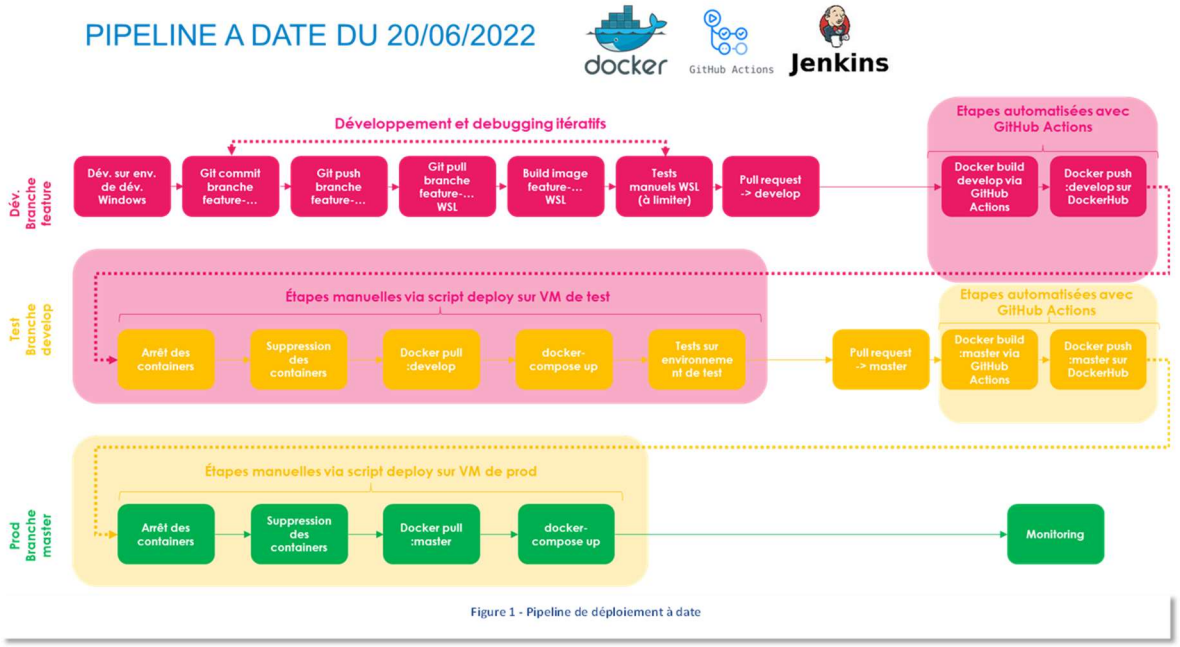

Figure n°5 : Pipeline de VPP actuel.

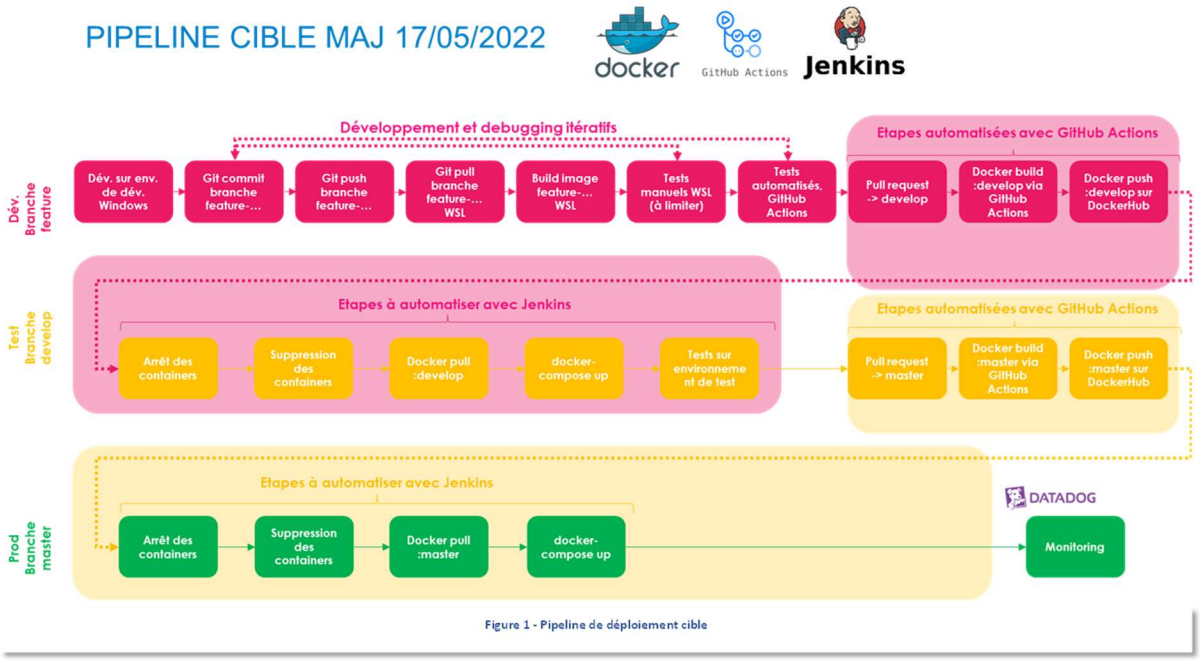

Figure n°6 : Pipeline de VPP cible.

b. Besoins de l’application HEX

A définir

c. Besoins de l’application de gestion de portefeuille

A définir

d. Besoins globaux

Les besoins globaux pour l’équipe des marchés de l’énergie sont les suivants :

- Unifier les différentes applications
- Mettre en place le CI/CD et les tests unitaires
- Confronter plusieurs outils pour déterminer les plus pertinents
- Mettre en place les bonnes pratiques de développement

BORALEX

Au-delà

DES ÉNERGIES RENOUVELABLES