

# Table des matières

<b>Remerciements</b>	<b>10</b>
<b>Résumé</b>	<b>11</b>
<b>1 Introduction Générale</b>	<b>1</b>
1.1 Contexte . . . . .	1
1.2 Problématique . . . . .	1
1.3 Objectifs . . . . .	2
1.4 Organisation du mémoire . . . . .	2
<b>I Étude De Préalable</b>	<b>4</b>
<b>2 Préambule</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Le Big Data . . . . .	5
2.2.1 Les Caractéristiques du Big Data . . . . .	5
2.3 Technologies Big Data . . . . .	5
2.3.1 Moteur d'analyse de grandes données : Apache Spark . . . . .	5
2.3.2 La plateforme HADOOP pour le calcul distribué de Big Data . . . . .	6
2.4 Technologies De Virtualisation . . . . .	6
2.5 La Technologie De Conteneurisation DOCKER . . . . .	6
2.5.1 Quels sont les différents éléments de Docker ? . . . . .	6
2.5.1.1 Docker Engine : . . . . .	7
2.5.1.2 Docker Daemon : . . . . .	7
2.5.1.3 Docker Client : . . . . .	7
2.5.1.4 Docker CLI . . . . .	7
2.5.1.5 Le registre Docker : . . . . .	7
2.5.2 Les avantages de Docker . . . . .	7
2.5.2.1 Assure La Portabilité . . . . .	7
2.5.2.2 Vitesse de déploiement . . . . .	7
2.6 L'orchestration De Conteneurs Kubernetes . . . . .	8
2.6.1 C'est Quoi Kubernetes ? . . . . .	8
2.6.2 Kubernetes, comment ça marche ? . . . . .	8
2.6.2.1 Contrôle Plane : Maître . . . . .	8
2.6.2.2 Worker/Node . . . . .	9
2.6.3 Avantages De Kubernetes . . . . .	9
2.6.3.1 Évolutivité . . . . .	9

## Table des matières

---

2.6.3.2	Haute disponibilité . . . . .	10
2.6.3.3	Sécurité . . . . .	10
2.6.3.4	Portabilité . . . . .	10
2.6.3.5	Self healing . . . . .	10
2.6.3.6	Scaling horizontal . . . . .	10
2.7	Conclusion . . . . .	10
<b>II</b>	<b>Analyse et Étude de l'existant</b>	<b>11</b>
<b>3</b>	<b>Présentation de l'organisme d'accueil</b>	<b>12</b>
3.1	Introduction . . . . .	12
3.2	Présentation . . . . .	12
3.3	Création et historique de la CASNOS . . . . .	13
3.4	Missions de la CASNOS . . . . .	13
3.4.1	Au niveau central : . . . . .	13
3.4.2	Au niveau de l'agence de wilaya : . . . . .	13
3.5	Organisation générale de la CASNOS . . . . .	14
3.6	Structure d'accueil . . . . .	14
3.6.1	Présentation de la direction de la modernisation et des systèmes d'Information . . . . .	14
3.6.2	Organigramme DMSI-CASNOS . . . . .	15
3.7	Conclusion . . . . .	15
<b>4</b>	<b>Analyse de l'existant</b>	<b>16</b>
4.1	Introduction . . . . .	16
4.2	LE RECUEIL DE L'INFORMATION . . . . .	16
4.2.1	Comment se fait le recueil de l'information pour l'étude de l'existant d'un projet ? . . . . .	16
4.2.2	Étude des moyens de traitement des informations . . . . .	16
4.2.2.1	Moyens matériels . . . . .	16
4.2.2.2	Moyens logiciels . . . . .	17
4.3	Présentation des applications CASNOS existantes . . . . .	17
4.4	Cartographie des Applications Métiers Principales de la CASNOS . . . . .	21
4.5	Critique de l'existant . . . . .	21
4.5.1	Les Problèmes trouvés dans les systèmes existants : . . . . .	21
4.6	Conclusion . . . . .	22
<b>III</b>	<b>Conception Et Modélisation De La Solution</b>	<b>23</b>
<b>5</b>	<b>Architectures de la plateforme</b>	<b>24</b>
5.1	Architecture fonctionnelle . . . . .	24
5.2	Introduction . . . . .	24
5.3	Présentation de l'architecture fonctionnelle . . . . .	24
5.4	Architecture Technique . . . . .	25
5.4.1	Qu'est ce qu'une architecture Technique ? . . . . .	25

## Table des matières

---

5.4.2	Justification de l'architecture technique . . . . .	27
5.4.3	Pourquoi cette séparation virtualisée ? . . . . .	27
5.4.4	Décortiquer l'architecture technique . . . . .	27
5.4.4.1	Noeuds Master . . . . .	27
5.4.4.2	Noeuds Workers . . . . .	29
5.5	Architecture Physique . . . . .	30
5.5.1	Explication de l'architecture physique . . . . .	30
5.5.2	Exigences minimales des technologies : . . . . .	30
5.6	Structure en couches de la plateforme . . . . .	31
5.6.1	Hardware . . . . .	32
5.6.2	Hyper-V . . . . .	32
5.6.3	Kubernetes Cluster . . . . .	32
5.6.4	SQL SERVER BIG DATA CLUSTER . . . . .	33
5.6.5	Services : Cas d'usages . . . . .	33
5.7	Conclusion . . . . .	33
<b>IV</b>	<b>Déploiement de la solution</b>	<b>34</b>
<b>6</b>	<b>Déploiement du cluster kubernetes</b>	<b>35</b>
6.1	Introduction . . . . .	35
6.2	Organisation des fichiers de commandes . . . . .	35
6.3	Pré-requis pour la configuration de Kubeadm . . . . .	36
6.4	Préparation des machines pour supporter le cluster Kubernetes . . . . .	36
6.5	Initialiser le cluster sur le noeud maître à l'aide de Kubeadm . . . . .	37
6.6	Joindre les noeuds workers . . . . .	38
6.7	Quelques commandes kubernetes . . . . .	39
6.7.1	Vérifier l'état du cluster . . . . .	39
6.7.2	Lister les pods du cluster . . . . .	40
6.8	Conclusion . . . . .	40
<b>7</b>	<b>Déploiement du big data cluster</b>	<b>41</b>
7.1	Introduction . . . . .	41
7.2	Étapes du déploiement . . . . .	41
7.3	Connexion au cluster . . . . .	44
7.3.1	Connexion à l'aide de AZDATA . . . . .	44
7.3.2	Afficher les endpoints du cluster . . . . .	44
7.4	Connecter le Cluster à AZURE DATA STUDIO . . . . .	45
7.4.1	AZURE DATA STUDIO, c'est quoi ? . . . . .	45
7.4.2	Se connecter au cluster . . . . .	45
7.5	Restaurer une base de données dans le cluster . . . . .	46
7.6	Vérifier l'état du cluster avec azure data studio . . . . .	48
7.7	Vérifier l'état du cluster avec kubernetes . . . . .	50
7.7.1	Explication de l'organisation des namespaces . . . . .	51
7.8	Conclusion . . . . .	51

## Table des matières

---

<b>V Présentation des cas d'usages</b>	<b>52</b>
<b>8 Apprentissage automatique</b>	<b>53</b>
8.1 Introduction . . . . .	53
8.2 L'apprentissage Automatique . . . . .	53
8.3 Les types d'apprentissage automatique . . . . .	53
8.3.1 Apprentissage supervisé . . . . .	54
8.3.1.1 Classification . . . . .	54
8.3.1.2 Régression . . . . .	55
8.3.2 Apprentissage non supervisé . . . . .	55
8.3.2.1 Clustering . . . . .	55
8.3.2.2 Réduction dimensionnelle . . . . .	56
8.3.3 Apprentissage par renforcement . . . . .	56
8.4 Principaux algorithmes du Machine Learning . . . . .	57
8.4.1 Arbres de décision . . . . .	57
8.4.2 Random forest . . . . .	58
8.4.3 XGBoost(eXtreme Gradient Boosting) . . . . .	60
8.5 Indicateurs de performances . . . . .	60
8.5.1 Précision . . . . .	61
8.5.2 La spécificité . . . . .	61
8.5.3 La sensibilité(Recall) . . . . .	61
8.5.4 L'exactitude (Accuracy) . . . . .	61
8.5.5 Le taux d'erreurs(error rate) . . . . .	61
8.5.6 F1-score . . . . .	62
8.5.7 ROC-AUC . . . . .	62
8.6 Conclusion . . . . .	62
<b>9 Conception</b>	<b>63</b>
9.1 Introduction . . . . .	63
9.2 Solution proposée . . . . .	63
9.3 Collecter les données . . . . .	64
9.4 Exploration des données . . . . .	65
9.5 Préparation des données . . . . .	67
9.6 Traitement des classes déséquilibrées . . . . .	68
9.7 Modèle de prédiction . . . . .	68
9.8 Le modèle optimale . . . . .	69
9.9 Conclusion . . . . .	69
<b>10 Implémentation</b>	<b>70</b>
10.1 Introduction . . . . .	70
10.2 Mise en place de l'environnement . . . . .	70
10.2.1 Visual Studio Code . . . . .	70
10.2.2 Anaconda . . . . .	70
10.2.3 Jupyter Notebook . . . . .	70
10.2.4 Python . . . . .	70
10.2.5 Numpy . . . . .	71
10.2.6 scikit-learn . . . . .	71

## Table des matières

---

10.2.7 Pandas . . . . .	71
10.3 Compréhension des données . . . . .	71
10.3.1 Importation des données . . . . .	71
10.4 Nettoyage des données . . . . .	71
10.5 Partition des données . . . . .	72
10.6 Évaluation . . . . .	72
10.7 Déploiement du modèle . . . . .	73
10.8 Conclusion . . . . .	75
<b>11 Autres cas d'usages</b>	<b>77</b>
11.1 Exécution des jobs spark sous SQL SERVER BIG Data Cluster . . . . .	77
11.1.1 Introduction . . . . .	77
11.1.2 Soumettre des travaux Spark sur le cluster Big Data SQL Server dans Azure Data Studio . . . . .	77
11.2 La virtualisation des données . . . . .	79
11.2.1 Cas d'utilisation Des Données Non Structurées : . . . . .	79
11.2.1.1 Étapes A suivre afin d'intégrer des données NOSQL (MongoDB) . . . . .	79
11.3 Exploration, visualisation et découverte des données logs avec Kibana . . . . .	80
11.4 Conclusion . . . . .	81
<b>VI Conclusion Et Perspectives</b>	<b>82</b>
11.5 Conclusion générale . . . . .	83
11.5.1 Présentation Du Future Système Informatique De La CASNOS Base Sur SQL SERVER Big Data . . . . .	84
11.6 Perspectives . . . . .	84
11.6.1 Déploiement Du Système D'information Et De Gestion D'assurance :OPENIMIS : . . . . .	84
11.6.1.1 Avantages OpenIMIS . . . . .	85
11.6.2 Envoi SMS en Temps Réel pour les usagers de la CARTE CHIFA . . . . .	85
11.6.3 L'ingestion de Données Logs et CDC à l'aide d'un cluster NIFI ET KAFKA . . . . .	85
<b>Annexes</b>	<b>87</b>
<b>A Préparation De L'environnement De Travail</b>	<b>88</b>
A.1 Introduction . . . . .	88
A.2 Préparation Du Serveur . . . . .	88
A.3 Création Des Machines Virtuelles via Hyper-V . . . . .	89
A.4 Conclusion . . . . .	92
<b>Annexes</b>	<b>88</b>
<b>B Choix Technologique</b>	<b>93</b>
B.1 Introduction . . . . .	93

B.2	Comment choisir la bonne technologie pour conduire une véritable transformation ? . . . . .	94
B.3	Les Technologies utilisées . . . . .	94
B.3.1	Sql Server 2019 . . . . .	94
B.3.2	Sql Server Big Data Cluster . . . . .	94
	B.3.2.1 Architecture SQL SERVER Big Data Cluster . . . . .	95
	B.3.2.2 Technologies mises en œuvre par le BDC . . . . .	95
	B.3.2.3 Pourquoi SQL SERVER BDC ? . . . . .	97
	B.3.2.4 Valeurs Ajoutées Pour La CASNOS : . . . . .	97
B.3.3	Apache HADOOP . . . . .	98
	B.3.3.1 Valeurs Ajoutées Pour La Casnos . . . . .	98
B.3.4	Apach Spark . . . . .	99
	B.3.4.1 Valeurs Ajoutées Pour La Casnos . . . . .	99
B.3.5	Docker . . . . .	100
	B.3.5.1 Valeurs Ajoutées Pour La Casnos . . . . .	100
B.3.6	Kubernetes . . . . .	101
	B.3.6.1 Valeurs Ajoutées Pour La Casnos . . . . .	101
	B.3.6.2 Kubernetes VS Docker Swarm . . . . .	102
	B.3.6.3 Popularité Kubernetes VS Docker Swarm . . . . .	103
	B.3.6.4 Récapitulatif De Comparaison . . . . .	104
B.3.7	Kubeadm . . . . .	105
	B.3.7.1 Pourquoi Kubeadm ? . . . . .	105
B.3.8	Grafana . . . . .	106
	B.3.8.1 Pourquoi Grafana ? . . . . .	106
B.3.9	Git . . . . .	107
B.3.10	Gogs . . . . .	107
	B.3.10.1 Pourquoi GOGS ? . . . . .	107
B.3.11	Windows Server 2019 -Datacenter- . . . . .	108
	B.3.11.1 Pourquoi Windows Server 2019 -Datacenter- ? . . . . .	108
B.3.12	Hyper-V . . . . .	109
	B.3.12.1 Pourquoi Hyper-V . . . . .	109
B.4	Conclusion . . . . .	109

## Table des figures

2.1	Architecture Docker . . . . .	6
2.2	Architecture Kubernetes . . . . .	8
3.1	Logo de la Caisse nationale de sécurité sociale des non-salariés (CASNOS) . . . . .	12
3.2	Organigramme générale de la CASNOS . . . . .	14
3.3	Organigramme DMSI-CASNOS . . . . .	15

## Table des figures

---

4.1	Compte rendu d'un interview pour le recueil d'information . . . . .	17
4.2	Schéma des Applications Métiers Principales de la CASNOS . . . . .	21
5.1	Architecture fonctionnelle proposée . . . . .	25
5.2	Architecture Technique Proposée . . . . .	26
5.3	Architecture Physique Proposée . . . . .	31
5.4	Couches de la plateforme . . . . .	32
6.1	Organisation Des Fichiers De configuration . . . . .	35
6.2	Setup-k8s-prereqs.sh . . . . .	36
6.3	Kubeadm Init . . . . .	37
6.4	Emplacement Du Fichier KubeConfig . . . . .	37
6.5	Plug-IN Réseau Flanenl . . . . .	37
6.6	Setup-k8s-master.sh . . . . .	38
6.7	Kubeadm Token Join . . . . .	38
6.8	Joindre Le Noeud Worker . . . . .	39
6.9	Vérifier L'état Du Cluster . . . . .	39
6.10	Lister Les Pods Du Cluster . . . . .	40
7.1	Installation Azure Data CLI . . . . .	41
7.2	Lancement de la création du BDC . . . . .	41
7.3	Choisir le type de déploiement du BDC . . . . .	42
7.4	Compte azdata du BDC . . . . .	42
7.5	Préciser la classe de stockage du BDC . . . . .	42
7.6	Création du BDC . . . . .	43
7.7	SQL SERVER BIG DATA CLUSTER Déployé . . . . .	44
7.8	Connexion Au Cluster . . . . .	44
7.9	Endpoints du cluster . . . . .	45
7.10	Se connecter au cluster via Azure Data Studio . . . . .	46
7.11	Connexion au cluster via Azure Data Studio Réussie . . . . .	46
7.12	Connexion au cluster via Azure Data Studio Réussie . . . . .	47
7.13	Copier le fichier de sauvegarde . . . . .	47
7.14	Le fichier de sauvegarde DBCASNOS.BAK . . . . .	47
7.15	Copier le fichier de sauvegarde . . . . .	47
7.16	DB CASNOS bien restaurée . . . . .	47
7.17	Requête d'affichage de DB CASNOS . . . . .	48
7.18	État globale du cluster avec Azure DS . . . . .	49
7.19	État globale du cluster avec GRAPHANA . . . . .	49
7.20	Azdata show statue . . . . .	50
7.21	Statu Globale Du Cluster . . . . .	50
7.22	Obtenir l'état des pods du cluster . . . . .	51
8.1	Les différentes types d'apprentissage automatique . . . . .	54
8.2	Exemple de classification . . . . .	55
8.3	Exemple de prédition de 5 clusters . . . . .	56
8.4	Illustration d'une arbre de décision . . . . .	57
8.5	Le processus d'une Forêt aléatoire . . . . .	59

## Table des figures

---

8.6 Bagging vs Boosting . . . . .	60
8.7 Matrice de confusion . . . . .	60
9.1 Description des colonnes de Dataset . . . . .	64
9.2 Le taux d'adhérent pour chaque classe . . . . .	65
9.3 Le taux d'adhérent pour chaque classe par catégorie . . . . .	66
9.4 Le taux d'adhérent pour chaque catégorie par classe . . . . .	66
9.5 Les valeurs nulles dans le jeu de données . . . . .	67
9.6 Les catégories des activités . . . . .	67
9.7 La date maximale de début de l'activité d'un adhérent . . . . .	68
10.1 Comparaison des résultats avant de d'appliquer SMOTE . . . . .	72
10.2 Comparaison des résultats après l'applicaton de SMOTE . . . . .	73
10.3 Comparaison des résultats pour les années 2020 et 2021 . . . . .	73
10.4 Surface d'accueil de Power Bi . . . . .	74
10.5 Fenêtre de récupération de données . . . . .	74
10.6 Interface de script python . . . . .	75
10.7 Déploiement du modèle sous forme d'un dashboard sur Power Bi . . . . .	75
11.1 Nouveau Spark Job . . . . .	77
11.2 Boîte de dialogue de soumission de Job Spark . . . . .	78
11.3 Surveiller la soumission des tâches Spark . . . . .	78
11.4 UI Spark job . . . . .	79
11.5 Data virtualization . . . . .	79
11.6 Choisir le type de donnees à virtualiser . . . . .	80
11.7 Création De Connexion Externe Avec Source De Données . . . . .	80
11.8 Log Spark Avec Kibana Et ElasticSearch . . . . .	81
11.9 Future Système Informatique De La CASNOS Base sur SQL SERVER Big Data . . . . .	84
11.10Logo OpenIMIS . . . . .	85
11.11Avantages OpenIMIS . . . . .	85
A.1 Gestionnaire Hyper-V . . . . .	88
A.2 Serveur Hyper-V . . . . .	88
A.3 Nouveau Ordinateur Virtuel Hyper-V . . . . .	89
A.4 Nommer la VM . . . . .	89
A.5 Spécifier la quantité de mémoire VM . . . . .	90
A.6 Choisir un réseau virtuel VM . . . . .	90
A.7 Choisir le disque dur virtuel dela VM . . . . .	91
A.8 Choisir Système D'exploitation Invité de la VM . . . . .	91
A.9 Se connecter à la VM . . . . .	91
A.10 Lancer l'installation Ubuntu à la VM . . . . .	92
B.1 La Transformation Digitale . . . . .	93
B.2 Logo SQL Server 2019 . . . . .	94
B.3 Logo SQL Server Big Data Cluster . . . . .	94
B.4 Architecture SQL Server Big Data Cluster . . . . .	95
B.5 Date Lake dans SQL Server Big Data Cluster . . . . .	96

## Table des figures

---

B.6 Polybase dans SQL Server Big Data Cluster . . . . .	96
B.7 AI/ML dans SQL Server Big Data Cluster . . . . .	97
B.8 Logo Apach Hadoop . . . . .	98
B.9 Logo Apach Spark . . . . .	99
B.10 Logo Docker . . . . .	100
B.11 Logo Kubernetes . . . . .	101
B.12 Docker Swarm VS Kubernetes . . . . .	102
B.13 Tableau Comparatif Kubernetes VS Docker Swarm . . . . .	103
B.14 Popularité Kubernetes VS Docker Swarm -Google Trends-	103
B.15 Kubeadm VS MiniKube VS AKS . . . . .	105
B.16 Grafana Logo . . . . .	106
B.17 Git Logo . . . . .	107
B.18 Gogs Logo . . . . .	107
B.19 Windows Server 2019 -Datacenter- Logo . . . . .	108
B.20 Hyper-V Logo . . . . .	109

# Remerciements

“ Tout d'abord, nous remercions Allah le tout puissant de nous avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

Nous tenons à remercier tout particulièrement notre encadrant Mr BOUGHEDDA Ahmed, pour l'aide compétente qu'il nous a apportée, pour sa patience et son encouragement. Son œil critique nous a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Nous tenons à remercier également notre promoteur BOUKHALFA pour son aide immense, la qualité de son suivie ainsi que pour tous les conseils et les informations qu'il nous a prodigués avec un degré de patience et de professionnalisme sans égal.

Nous tenons aussi à adresser nos plus sincères remerciements à Mr Mohamed Ala AL CHIKHA et Mme Naouel BOUDAR pour leurs aide et leurs disponibilités. Ainsi que toutes l'équipe de CASNOS pour leurs aides et renseignements précieuses, qu'ils nous ont fournis ainsi que pour leurs encouragements et pour avoir rendu notre stage à de PFE une expérience très enrichissante.

Que les membres de jury trouvent, ici, l'expression de nos sincères remerciements pour l'honneur qu'ils nous font en prenant le temps de lire et d'évaluer ce travail.

Pour finir, nous souhaitons remercier toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

”

# Résumé

Dans le cadre de la réalisation de son plan d'action 2021-2026, la CASNOS vise à améliorer la qualité de services vis-à-vis de ses usagers (Adhérents, assurés sociaux et retraités) via l'externalisation de ces derniers (e-services).

Les responsables du département informatique de l'organisme pensent à une solution digitale intégrée qui permettra de faciliter la prise de décision et d'orienter le développement de ses futures activités.

Pour ce faire, il est indispensable de mettre en œuvre une plateforme BIG DATA intégrée, robuste et puissante pour le traitement des requêtes OLTP en cas de remonté en charge tout en offrant un délai d'attente réduit. En outre, sur le volet analytique (OLAP), la plateforme digitale doit avoir aussi la capacité à livrer une valeur ajoutée sur le plan analyse de données pour une meilleure prise de décision (Business intelligence moderne, machine learning pour l'actuariat et détection de fraude, etc.). Enfin, sur le plan infrastructure, la plateforme doit être évolutive (scalable et maintenable) afin de se réajuster à l'évolution interne et externe de l'organisme.

Pour conclure, le but initial de ce projet de fin de cycle est de contribuer à la concrétisation de cette stratégie via : 1-La proposition d'une architecture big-data Lambda/kappa orientée Micro-services (Docker,Kubernetes) et les technologies Microsoft (SQL Server 2019) 2-Aligner l'écosystème BIG-DATA à un cas d'usage (Tableau de bord temps-réel, envoie SMS, etc.) 3-Introduction de la méthode DEVOPS au sein de l'équipe IT de la CASNOS.

# **Chapitre 1**

## **Introduction Générale**

Dans cette première partie nous aborderons le contexte thématique du projet, les enjeux impliqués, les objectifs et enfin nous détaillerons l'aspect organisationnel de ce mémoire.

### **1.1 Contexte**

La Caisse Nationale de Sécurité Sociale des Non-Salariés - CASNOS est chargée de la protection sociale des catégories professionnelles non-salariées pour les risques suivants : les assurances relatives à la maladie et à la maternité, l'assurance invalidité, l'assurance décès et l'assurance vieillesse. Elle s'en charge du paiement de toutes les prestations, contrôle médical , la gestion de la pension et l'allocation de retraite des assurés et leurs ayants droit. Par conséquent, nous trouvons plusieurs applications métiers dédiées à l'accomplissement des missions principales de la Caisse.

Dans la poursuite de sa politique de modernisation des services, la CASNOS souhaite apporter de nombreuses réformes touchant de multiples aspects. Outre le lancement d'un système d'envoie d'SMS en masse en temps réel pour les bénéficiaires de la Carte CHIFA, ou encore le monitoring en temps réel de la réPLICATION. L'entreprise cherche à apporter une réforme au système actuelle ; en touchant à la fois à son fonctionnement et à son aspect technique, afin d'améliorer ses performances et sa précision, et l'adapter à une montée en charge dans les perspectives d'une augmentation de la volumétrie des données, et dans le but de proposer un service rapide, de qualité et up-to-date d'un point de vue technique.

### **1.2 Problématique**

Dans le cadre de la réalisation de son plan d'action 2021-2026, la CASNOS vise à améliorer la qualité de services vis-à-vis de ses usagers (Adhérents, assurés sociaux et retraités) via l'externalisation de ces derniers (e-services).

La situation actuelle des systèmes de la CASNOS et l'utilisation quotidienne des services numériques de l'organisme a rendu la production de données et de l'information très riche, a un flux très important avec une nécessité de disponibilité très élevée. la CASNOS présente actuellement plusieurs problèmes parmi lesquels nous citons :

- Des systèmes distribuées qui augmentent le nombre d'accès simultané aux bases de données sources ce qui engendrent des workload.
- La charge de travail issue des accès concurrents fait en sorte d'affaiblir les performances des systèmes.
- Des systèmes décentralisés ce qui engendrent une difficultés dans la conception des solutions futures.
- Une forte augmentation du volume de données générées et stockées, ce qui entraîne un problème de redondance, de manipulation et surtout d'exploitation de ces données.
- La croissance du nombre de besoins d'analyse pose un problème de vélocité.
- Parfois le temps de latence est important surtout quand des transaction (Update , Insert) sont appliquées sur des tables de volumétrie importantes (37 Millions de Lignes).
- Les tableaux de bord ne sont pas en temps réels.

### 1.3 Objectifs

Afin de résoudre la problématique exposée précédemment, nous avons opté pour l'alignement d'un écosystème de type Big Data basée sur SQL SERVER 2019 aux attentes et besoins de l'organisme pour une meilleure gestion et analyse des données, tout en mettant en place comme cas d'usage d'interrogation de donne avec Apache Spark , de plus un modèle de Machine Learning.

Pour ce faire, il est indispensable de mettre en œuvre une plateforme BIG DATA intégrée, robuste et puissante pour le traitement des requêtes OLTP en cas de remonté en charge tout en offrant un délai d'attente réduit. En outre, sur le volet analytique (OLAP), la plateforme digitale doit avoir aussi la capacité à livrer une valeur ajoutée sur le plan analyse de données pour une meilleure prise de décision (Business intelligence moderne, machine Learning pour l'actuariat et détection de fraude, etc.). Enfin, sur le plan infrastructure, la plateforme doit être évolutive (scalable et maintenable) afin de se réajuster à l'évolution interne et externe de l'organisme.

### 1.4 Organisation du mémoire

Afin de répondre à notre objectif, notre travail est structuré ainsi :

- **Partie 1 : Background** Ce chapitre vise à présenter les notions et concepts relatifs à notre projet, nous avons commencé d'abord par définir le domaine de Big Data, ses caractéristiques et ses technologies . Ensuite nous allons définir la virtualisation et conteneurisation des données ainsi que ses outils.
- **Partie 2 : Analyse de l'existant** Nous présenterons dans ce chapitre, l'organisme qui nous a accueilli « CASNOS », ainsi que le département informatique dans lequel nous avons évolué tout au long de la durée du stage.

- **Partie 03 : Conception technique de la solution** Dans cette partie nous allons entamer la présentation de notre solution, en passant par nos architectures proposées.
- **Partie 4 : Déploiement de la solution** À travers ce dernier chapitre, nous citerons les différentes étapes suivies pour la réalisation et l'évaluation du nouveau système et parlerons des diverses technologies et outils utilisés
- **Partie 5 : Présentation Des Cas D'usages** Dans ce chapitre nous présenterons les cas d'usage ainsi qu'un aperçu sur les scénario possible afin d'exploiter les performance de la plaeforme
- **Partie 6 : Conclusion** Nous clôturons notre mémoire par une conclusion générale ainsi que des perspectives de notre projet.

# Première partie

## Étude De Préalable

# Chapitre 2

## Préambule

### 2.1 Introduction

Avec le pic d'utilisation d'internet et l'évolution de la technologie, l'utilisation des données a augmenté ces dernières années, rapidement de façon exponentielle dans tous les domaines comme le marketing, la météorologie, le trafic, la santé, etc. Dans ce chapitre, nous présenterons le Big Data et ses caractéristiques ainsi les technologies utilisées pour la réalisation du projet.

### 2.2 Le Big Data

Le terme big data est apparu au début des années 90 qui désigne les « méga-données» ou jeux de données volumineux et complexes ( structurées ou non-structurées : texte, audio, vidéo.etc) qui se génèrent quotidiennement par différentes sources avec une grandes vitesses.

#### 2.2.1 Les Caractéristiques du Big Data

Le Big Data possède des caractéristiques et des attributs spécifiques qui peuvent nous aider à comprendre à la fois les défis et les avantages du Big Data.

- Le Volume
- La vitesse
- La Variété
- La Variabilité
- La Véracité
- La Validité
- La Vulnérabilité
- La Volatilité
- La Visualisation
- La Valeur

### 2.3 Technologies Big Data

#### 2.3.1 Moteur d'analyse de grandes données : Apache Spark

Spark est un environnement open source dédié au calcul distribué à grande échelle construit autour de la vitesse, de la facilité d'utilisation et d'analyses sophistiquées. Il est essentiellement dédié au Big Data et Machine Learning.

### 2.3.2 La plateforme HADOOP pour le calcul distribué de Big Data

Hadoop est un projet Open Source géré par Apache Software Fundation basé sur le principe Map-Reduce. Le principe repose sur l'exécution du traitement répartie multi-noeuds pour augmenter drastiquement les capacités de calculs et de stockage afin de traiter de très grandes quantités de données.

Son système de fichiers distribué favorise un taux élevé de transfert de données entre les noeuds et permet un fonctionnement ininterrompu du système en cas de défaillance ou de panne d'un d'entre eux. Cette approche diminue le risque de panne majeure, même lorsqu'un nombre important de noeuds deviennent inopérants.

## 2.4 Technologies De Virtualisation

La virtualisation est le processus d'exécution d'une instance virtuelle d'un système informatique sur une couche d'abstraction sur le matériel réel. Le plus souvent, cela fait référence à l'exécution simultanée de plusieurs systèmes d'exploitation sur un système informatique.

## 2.5 La Technologie De Conteneurisation DOCKER

Docker est une plate-forme de conteneurs qui a été introduite en 2013 et a contribué à démocratiser la conteneurisation. Créer facilement des conteneurs et des applications basées sur des conteneurs. Il en existe d'autres, mais celui-ci est le plus couramment utilisé. Il est également plus facile à déployer et à utiliser que ses concurrents.

### 2.5.1 Quels sont les différents éléments de Docker ?

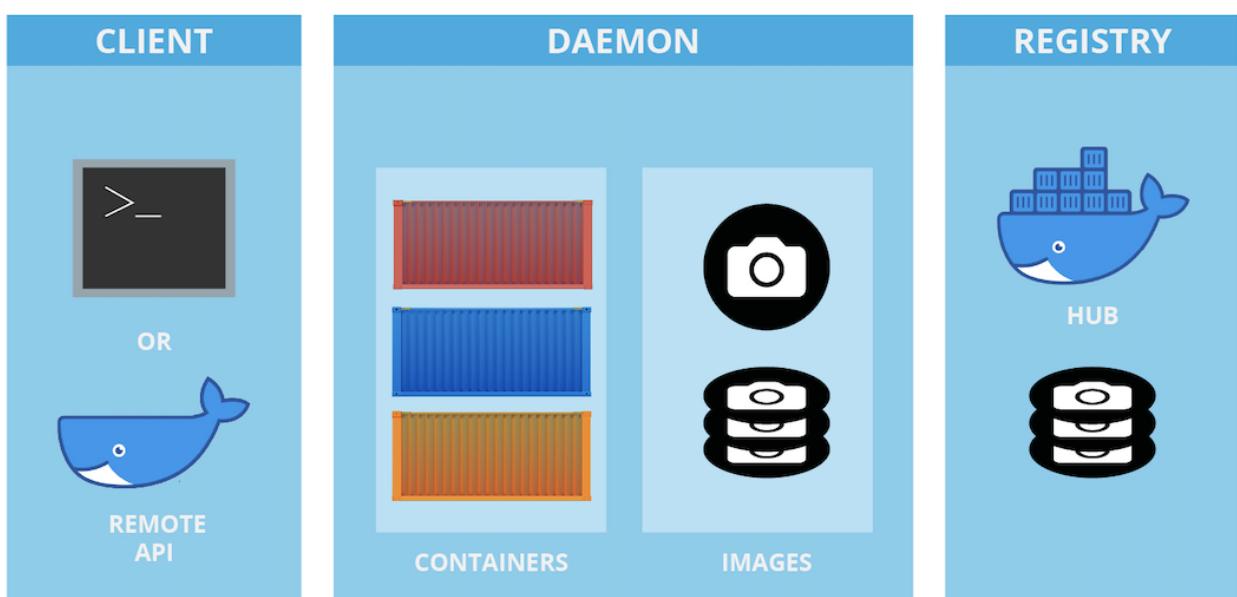


FIGURE 2.1 – Architecture Docker

### 2.5.1.1 Docker Engine :

Docker Engine est le système qui fait tourner l'ensemble de Docker. Il s'agit d'un moteur qui suit une architecture client-serveur comprenant principalement trois composants : Docker Daemon, Docker CLI et une API qui sert entre autres d'intermédiaire entre les deux composants précédents. Elle permet de créer, exécuter et gérer des conteneurs Docker. En tant que cœur du système Docker, il réunit tous les composants de la plate-forme en un seul endroit.

### 2.5.1.2 Docker Daemon :

Cheval de bataille du système Docker, ce composant écoute et traite les demandes d'API pour gérer les divers autres aspects de votre installation, tels que les images, les conteneurs et les volumes de stockage. Docker Daemon peut également communiquer avec d'autres Docker Daemon afin de gérer les services.

### 2.5.1.3 Docker Client :

Il s'agit de l'interface utilisateur pour communiquer avec le système Docker. Il accepte les commandes via l'interface de ligne de commande (CLI) et les envoie au démon Docker.

### 2.5.1.4 Docker CLI

Docker CLI est l'abréviation de Docker Command Line Interface. Il s'agit des commandes à exécuter lors de chaque opération que l'on effectue sur Docker. Il existe trois types de CLI.

### 2.5.1.5 Le registre Docker :

Un catalogue pour héberger, pousser et extraire des images Docker. Il est possible d'utiliser un registre local ou l'un des nombreux services de registre hébergés par des tiers(Tel :Red Hat Quay, Docker Hub).Un registre Docker organise les images dans des emplacements de stockage, appelés référentiels , où chaque référentiel contient différentes versions d'une image Docker qui partagent le même nom d'image.

## 2.5.2 Les avantages de Docker

### 2.5.2.1 Assure La Portabilité

puisque les applications n'ont pas à être liées au système d'exploitation hôte. Les applications conteneurisées peuvent par exemple être aisément transférées de systèmes sur site vers les environnements Cloud.

### 2.5.2.2 Vitesse de déploiement

Docker aide à augmenter la vitesse de déploiement de plusieurs mois à plusieurs semaines. Nous pouvons facilement déployer un nouveau code en production en l'intégrant au CI/CD.

## 2.6 L'orchestration De Conteneurs Kubernetes

### 2.6.1 C'est Quoi Kubernetes ?

Kubernetes est un projet open Source permettant de déployer, faire évoluer et gérer des applications conteneurisées, sur n'importe quelle plate-forme.

Il offre la possibilité d'automatiser le déploiement et la gestion à grande échelle des applications multi-conteneurs. Il s'agit d'un système permettant d'exécuter et de coordonner des applications conteneurisées sur un cluster de machines. Possède une plate-forme conçue pour gérer entièrement le cycle de vie des applications et des services conteneurisés en utilisant des méthodes de prévisibilité, d'évolutivité et de haute disponibilité.

### 2.6.2 Kubernetes, comment ça marche ?

Voici le schéma général de l'architecture de Kubernetes. Il est composé de deux grandes parties : le master et le worker (ou node).

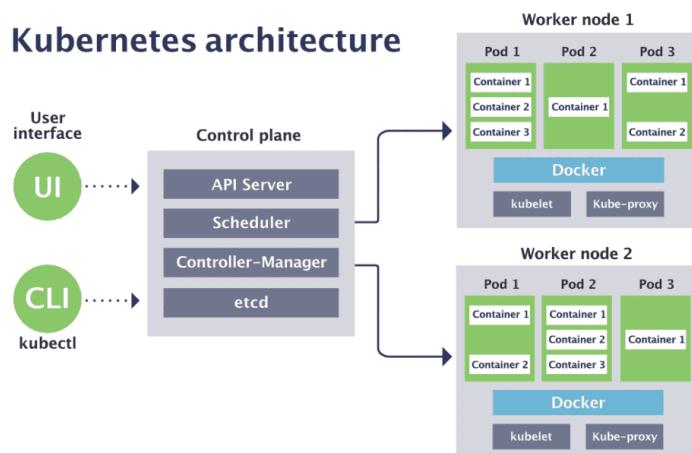


FIGURE 2.2 – Architecture Kubernetes

#### 2.6.2.1 Contrôle Plane : Maître

Le maître gère la planification et le déploiement des instances d'application sur les noeuds, et l'ensemble complet de services exécutés par le noeud maître est appelé plan de contrôle. Le maître communique avec les noeuds via le serveur d'API Kubernetes . Le planificateur attribue des noeuds aux pods (un ou plusieurs conteneurs) en fonction des contraintes de ressource et de stratégie que vous avez définies.

**Le Master est composé de plusieurs parties :**

1. **kube-apiserver** Le serveur d'API Kubernetes est l'entité de gestion centrale qui reçoit toutes les demandes REST de modifications (aux pods, services, ensembles de réPLICATION/contrôleurs et autres), servant d'interface au cluster. De plus, c'est le seul composant qui communique avec le cluster etcd, s'assurant que les données

sont stockées dans etcd et sont en accord avec les détails de service des pods déployés.

2. **kube-scheduler** Aide à planifier les pods sur les différents nœuds en fonction de l'utilisation des ressources. Il lit les exigences opérationnelles du service et le planifie sur le nœud le mieux adapté.

**Exemple :** si l'application a besoin de 1 Go de mémoire et de 2 cœurs de processeur, les pods de cette application seront planifiés sur un nœud avec au moins ces ressources. Le planificateur s'exécute chaque fois qu'il est nécessaire de planifier des pods. Le planificateur doit connaître le total des ressources disponibles ainsi que les ressources allouées aux charges de travail existantes sur chaque nœud.

3. **kube-controller-manager**

Exécute un certain nombre de processus de contrôleur distincts en arrière-plan pour réguler l'état partagé du regrouper et effectuer des tâches de routine. Lorsqu'un changement dans la configuration d'un service se produit

4. **Etcd** un stockage de valeur de clé simple et distribué qui est utilisé pour stocker les données du cluster Kubernetes (telles que le nombre de pods, leur état, l'espace de noms, etc.), les objets API et les détails de découverte de service. Il n'est accessible que depuis le serveur API pour des raisons de sécurité.

### 2.6.2.2 Worker/Node

Les clusters sont constitués de nœuds, chacun représentant un seul hôte informatique, c'est-à-dire une machine virtuelle ou physique. Les noeuds exécutent les tâches qui leur sont confiées, sous la direction du noeud maître qui les contrôle. Les principaux composants trouvés sur un nœud (worker) :

1. **kubelet** Le service principal sur un nœud worker, prenant régulièrement en compte les spécifications de pod nouvelles ou modifiées (principalement via le kube-apiserver) et s'assurant que les pods et leurs conteneurs sont sains et fonctionnent dans l'état souhaité.
2. **kube-proxy** un service proxy qui s'exécute sur chaque nœud de travail pour gérer les sous-réseaux d'hôtes individuels et exposer les services au monde extérieur. Il effectue le transfert des demandes vers les pods/conteneurs appropriés sur les différents réseaux isolés d'un cluster.
3. **Container RunTime :Docker** Vu que les Pods sont des conteneurs .Un système de conteneurisation est nécessaire .Il existe plusieurs choix tel : Docker , ContainerD etc .

### 2.6.3 Avantages De Kubernetes

#### 2.6.3.1 Évolutivité

Kubernetes fournit une mise à l'échelle horizontale des pods sur la base de l'utilisation du processeur. Le seuil d'utilisation du processeur est configurable et Kubernetes démarrera automatiquement de nouveaux pods si le seuil est atteint.

### **2.6.3.2 Haute disponibilité**

Kubernetes offre une haute disponibilité tant au niveau de l'application qu'au niveau de l'infrastructure.

### **2.6.3.3 Sécurité**

Kubernetes traite la sécurité à plusieurs niveaux : cluster, application et réseau. Les points de terminaison de l'API sont sécurisés via la

### **2.6.3.4 Portabilité**

La portabilité de Kubernetes se manifeste en termes de choix de système d'exploitation, d'architectures de processeur (machines virtuelles ou bare metal), de fournisseurs de cloud (AWS, Azure ou GCP) .

### **2.6.3.5 Self healing**

Kubernetes remplace et replanifie automatiquement les conteneurs défaillants. Les conteneurs qui échouent à une vérification de l'état sont arrêtés et redémarrés. Il empêche également le trafic d'être acheminé vers des conteneurs indisponibles.

### **2.6.3.6 Scaling horizontal**

Les applications peuvent être mises à l'échelle manuellement ou automatiquement en fonction du processeur/de la mémoire ou de métriques personnalisées.

## **2.7 Conclusion**

Dans ce chapitre nous avons présenté les concepts de base du Big Data ainsi que les technologies de conteneurisation. Nous allons entamer sur les prochains chapitres, les différents concepts restants autour de notre projet.

## Deuxième partie

# Analyse et Étude de l'existant

# Chapitre 3

## Présentation de l'organisme d'accueil

### 3.1 Introduction

Le but envisagé par cette partie est de faire une présentation globale de notre organisme d'accueil **CASNOS** (Caisse nationale d'Assurance Sociale des Non-salariés) d'ALGER -Direction Generale- où nous avons effectué notre stage pratique.

### 3.2 Présentation

La sécurité sociale est la protection qu'une société offre aux individus de faire face aux conséquences financières des risques sociaux tel : vieillesse, maladie, invalidités .etc  
Les personnes susceptibles d'y être assujetties sont les suivantes : Les artisans, les membres de professions libérales , les associés ou gérants , les artistes payés au cachet, les agriculteurs, les commerçants.

La CASNOS est chargée de la protection sociale des catégories professionnelles non-salariées pour les risques suivants :

- les assurances relatives à la maladie et à la maternité.
- l'assurance invalidité, l'assurance décès.
- l'assurance vieillesse.



FIGURE 3.1 – Logo de la Caisse nationale de sécurité sociale des non-salariés (CASNOS)

### **3.3 Creation et historique de la CASNOS**

Au lendemain de l'independance (1962), en Algrie existait un systme de scurit sociale trs fragment, compos de plus de 11 diverses rgimes : des caisses de rgimes spciaux, rgime agricole, de secours minier, caractriss par des diffrences dans le financement, la nature et le niveau des prestations ainsi que le mode de gestion.

C'est  partir des annes 1970 que la problmatique de la rforme du systme de scurit sociale est pose. Les buts de la rforme etaient l'unification des rgimes, l'uniformisation des avantages et l'extension des bnficiaires . Des amliorations importantes ont t alors apportes par voie des circulaires au niveau des prestations servies.

En 1988 le statut juridique des deux caisses fut modfi en Etablissement Public  caractre Spcifique,  la faveur de la loi 88-01 qui porta l'autonomie aux entreprises publiques.

En 1992 la CNASAT changeait d'appellation par Caisse Nationale des Assurances Sociales des Travailleurs Salaris (C.N.A.S.), et il y avait l'institution de la Caisse des Assurances Sociales des Non Salaris (C.A.S.N.O.S.). [2]

### **3.4 Missions de la CASNOS**

Dans le cadre des lois et rglements en vigueur, la caisse a pour mission :

#### **3.4.1 Au niveau central :**

La direction gnrale de la caisse est charge notamment de :

- Organiser, coordonner et contrôler les activites des agences de wilaya et des antennes, ainsi que la gestion des moyens humains et matriels de la caisse.
- Mettre en application les dispositions relatives  toutes les prestations prvues par les conventions et accords internationaux.
- Normaliser les mthodes de gestion.
- Mettre en place un systme d'information en mesure de fournir aux responsables de la caisse l'lment indispensable  la matrise du systme et au suivi de son dveloppement.
- Mener une politique financire et mme d'assurer l'quilibre financier du systme.

#### **3.4.2 Au niveau de l'agence de wilaya :**

Dans le souci de rapprocher l'institution du citoyen, les agences et antennes sont charges des tches relatives  : L'affiliation des assurs.

- Au paiement de toutes les prestations
- Au contrle mdical
- La gestion de la pension et l'allocation de retraite des assurs et leurs ayants droit.

## 3.5 Organisation générale de la CASNOS

L'organisation générale de la CASNOS est constituée de ce que suit :

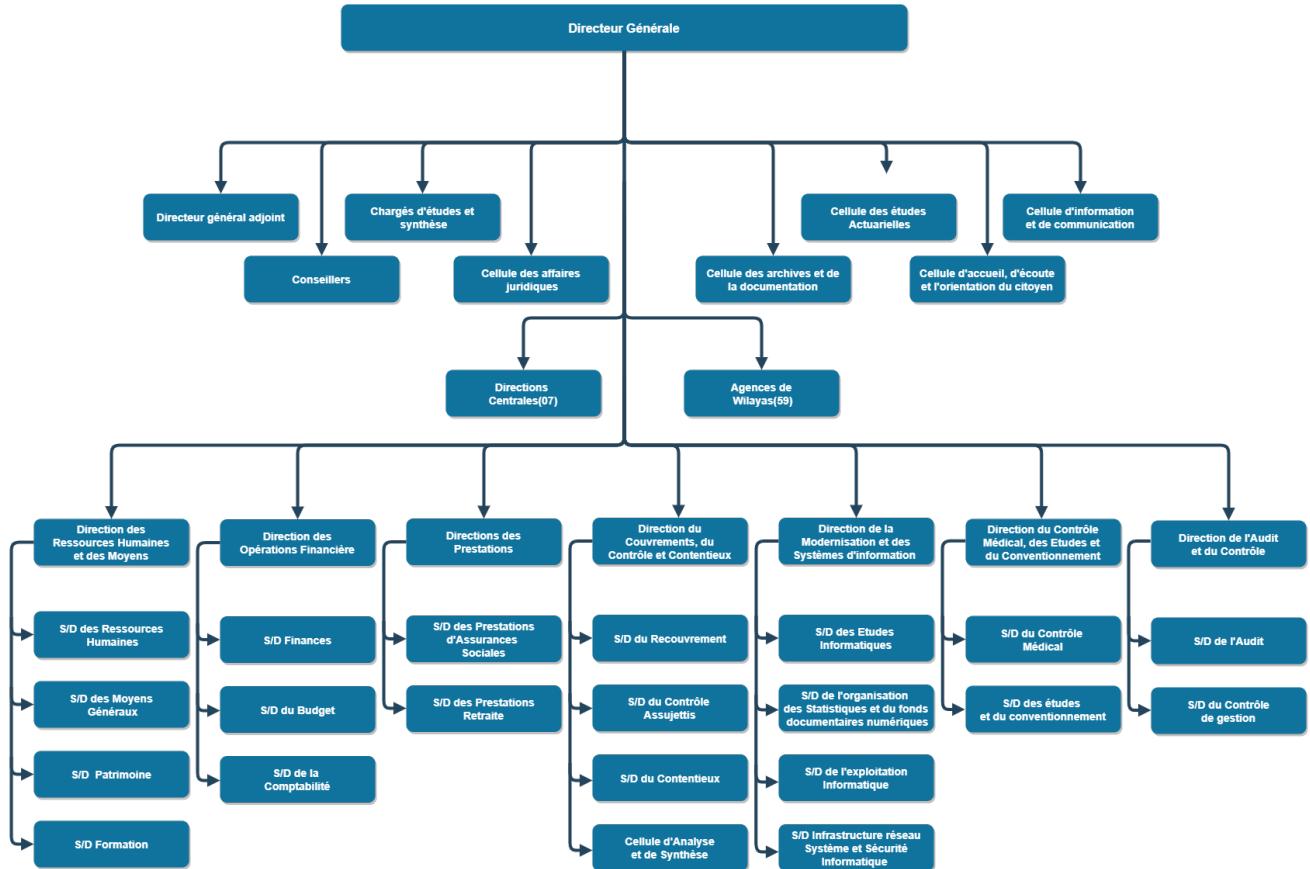


FIGURE 3.2 – Organigramme générale de la CASNOS

## 3.6 Structure d'accueil

### 3.6.1 Présentation de la direction de la modernisation et des systèmes d'Information

Nous avons réalisé notre PFE au niveau de direction de la modernisation et des systèmes d'information. Elle comprend quatre (04) sous-directions :

- La sous-direction des études informatiques ;
- La sous-direction de l'organisation, des statistiques et du fonds documentaire numérique.
- La sous-direction de l'exploitation informatique ;
- La sous-direction infrastructure réseau, système et sécurité informatiques.

## Chapitre 3. Présentation de l'organisme d'accueil

La Direction de la modernisation et des systèmes d'information est chargée de :

- D'assurer la modernisation des systèmes d'information et des systèmes de gestion de la caisse.
- D'assurer l'assistance des agences de wilaya en matière de système d'information.
- D'assurer la sécurité informatique de la caisse et élaborer les tableaux de bord de gestion.
- D'administrer et de suivre les portails électroniques, le site WEB et le réseau Internet.

### 3.6.2 Organigramme DMSI-CASNOS

Elle comprend les quatre (04) sous-directions suivantes ainsi que leurs missions :

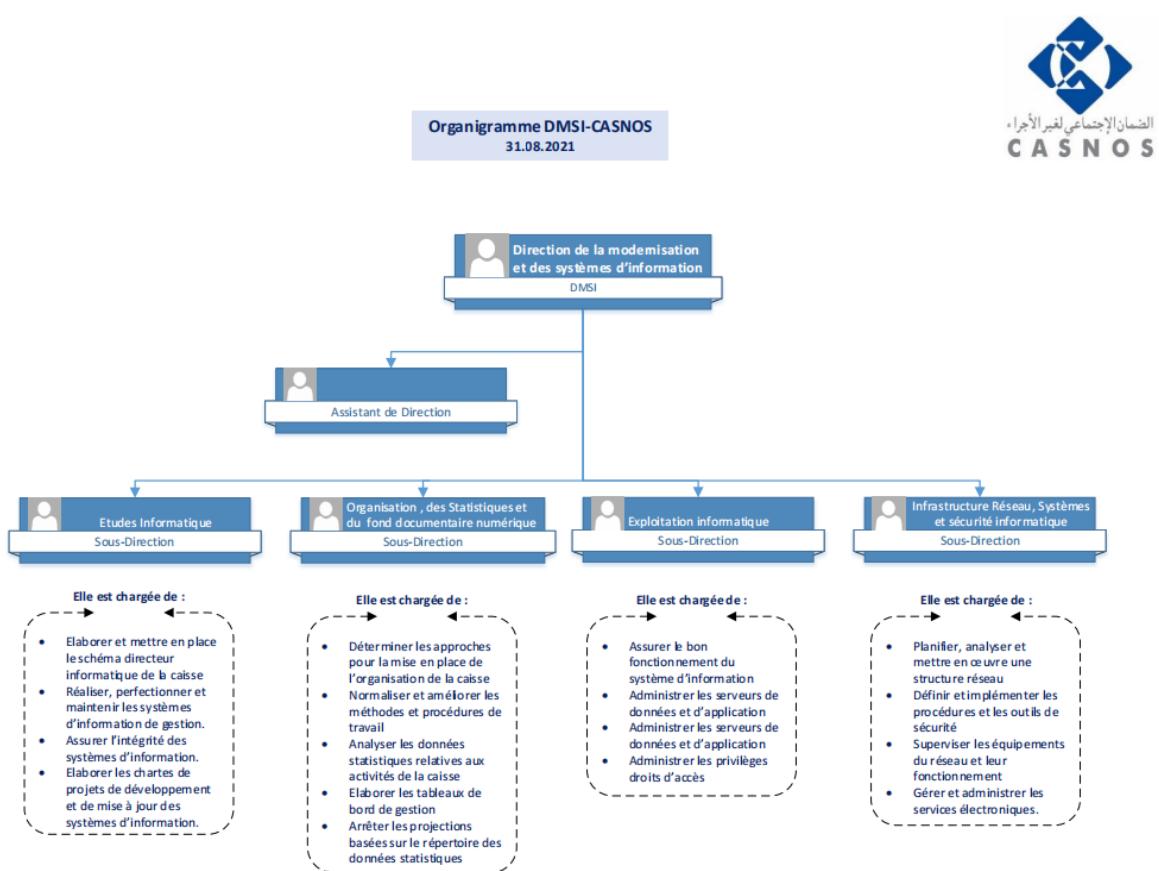


FIGURE 3.3 – Organigramme DMSI-CASNOS

## 3.7 Conclusion

Dans ce chapitre nous avons présenté la Caisse Nationale de Sécurité Sociale des Non-Salariés- CASNOS qui a accueilli notre projet de fin d'études, en identifiant ses missions et son organisation interne.

# Chapitre 4

## Analyse de l'existant

### 4.1 Introduction

L'analyse de l'existant permet de comprendre la nature du système actuel, décrit la solution présente du domaine d'étude au terme d'organisation. Le but de l'analyse de l'existant est la recherche des points forts et des points faibles du système existant. Ainsi, l'analyse de l'existant fait l'état du système actuel.

### 4.2 LE RECUEIL DE L'INFORMATION

Avant de se lancer dans un projet, une analyse de l'existant s'impose. Il n'est donc pas question de s'aventurer sans avoir élaboré une analyse complète des projets ou des concurrents.

#### 4.2.1 Comment se fait le recueil de l'information pour l'étude de l'existant d'un projet ?

L'analyse de l'existant se fait en utilisant plusieurs méthodes :

- **Interviews** : Consiste à établir des interviews ou réunions avec les experts de l'entreprise afin de comprendre le métier et de découvrir les applications existantes
- **Questionnaires** : Préparer des questions en ce qui concerne : Les logiciels classiques utilisés, les bases de données, la sauvegarde, l'archivage, la sécurité informatique .etc
- **Lecture de documentation** : Catalogues, études, données statistiques etc )

#### 4.2.2 Étude des moyens de traitement des informations

##### 4.2.2.1 Moyens matériels

L'entreprise dispose en son sein d'un grand nombre d'ordinateurs, des serveurs dédiés, des imprimantes, des onduleurs, des stabilisateurs et des outils disponible pour le réseau (Switch, routeur).

## Chapitre 4. Analyse de l'existant

### 4.2.2.2 Moyens logiciels

Les logiciels utilisés sont : Microsoft office, pour les éventuelles saisies , SQL Server pour la gestion de base de données , Windows Server et autre logiciels.

Afin d'illustrer la phase de la récolte d'information, ci joint un exemple de compte rendu de l'interview avec *la sous directrice système et réseau de la DMSI* :

COMPTE-RENDU DE L'ENTRETIEN N-01				
 Lieu : Casnos ALGER العنوان: ٦٣ شارع العروبة - برج الورش - الدار البيضاء - المغرب CASNOS	NOM DU PROJET : "Conception et implémentation d'une plate-forme Big-Data intégrée dans le cadre de la stratégie de transformation Digitale"	Date de la réunion :	25/07/2022 Heure : 10.00-10.30	
<b>NOM</b>	Entreprise/Role			
Zahia Messahel	Sous directrice système et réseau			
AGOUDJIL Katia	Étudiante			
LADLANI Randa	Étudiante			
<b>Questions</b>	<b>Réponses</b>			
1/ Présentation générale du département système et réseau				
2/ Réponses aux questions dans le cadre de l'étude de l'existant				
<b>Le réseau local du projet</b>				
• Le réseau local est-il sans fil ou câblé ou mixte ?	câblé			
• Le réseau est-il installé conformément aux normes ?	Oui et ses normes sont respectées dans tous les sous systèmes , agences etc			
• Le réseau est-il suffisamment sécurisé ?	Limiter les accès Internet, Imposer un VPN pour l'accès à distance etc			
• Le réseau est-il suffisamment perçu comme performant par les utilisateurs ?	Oui			
• Quels sont les usages courants et spéciaux du système réseau dans le projet	Ingénieries			
<b>La connectivité Internet</b>				
• Le mystère est-il connecté à Internet ?	Oui , mais non ouvert			
• Qui sont connectés à Internet ? et comment se fait la connexion ?	Au besoin , en utilisant un câble + accès proxy ≈6Mbps			
• Quelle est le débit et la qualité de la connexion Internet ?				
<b>Les systèmes d'exploitation utilisés</b>	90% Windows et 10% Linux			
<b>La sécurité informatique</b>				
• Le projet possède-t-il une politique de sécurité informatique pour prévenir tout anomalie	Oui			
• Sécurité des données et fichiers	Notion de Privileges			
• Sécurité d'accès physique et logiciel aux postes de travail	Poste de travail sécurisé			
• Hiérarchie de sécurité d'accès aux données et fichiers	Au besoin +privileges			
• Sécurité contre les virus	L'installation d'un pare-feu et d'un antivirus.			
• Restriction sur les Mots de Passe	Oui, respectant les normes : Plus de 8 caractères alphanum etc			
<b>la gestion d'accès</b>				
• Le mystère a-t-il une politique d'optimisation et la gestion d'accès ?	Oui a l'aide d'un mystère annuaire : Chacun son compte , pas de compte commun			
• Le mystère a-t-il une politique pour la gestion d'accès à la données	Oui , chaque user a son propre compte + les priviléges			
• Politique pour la gestion de SQL SERVER	System annuaire + MDP à la norme			
<b>Note</b>				
Vu que tout est centralisé => Toutes les restrictions + Containtes appliquées au niveau local (Casnos ) System/APP/BD sont aussi appliquées au niveau des agences et sur le terrain c'est une obligation				
<b>Conclusions</b>				
Pas de problèmes a signalé car pour l'instant tout est conforme aux normes				

FIGURE 4.1 – Compte rendu d'un interview pour le recueil d'information

## 4.3 Présentation des applications CASNOS existantes

En ce qui suit , nous allons présenter les principales applications financières critiques de la CASNOS ; utilisateurs, fonctionnalités .etc

1. Syscas
2. Plateforme AS
3. SYSRET
4. SIAD
5. Damankoum

## Chapitre 4. Analyse de l'existant

---

Les Principales Applications Métiers					
Application	Utilisateurs	Fonctionnalités	Date de Mise en place	Volume	Nombres d'accès
Syscas	Contrôle de Gestion	<ul style="list-style-type: none"> <li>- Gestion des encaissements</li> <li>- Gestion des cotisants</li> <li>- Gestion de la comptabilité recouvrement</li> <li>- Édition des bilans</li> <li>- Gestion du Contentieux</li> <li>- Gestion des mises en demeure</li> <li>- Module pour la gestion du Contrôle Cotisant</li> </ul>	01/07/2011	120G en croissance	4779
Plateforme AS	Contrôle de Gestion	<ul style="list-style-type: none"> <li>- Gestion et mise à jour des couvertures de droit(Assuré et ses ayants droit)</li> <li>- Gestion des professionnels de la santé (Pharmacies, Médecins, Endimed, structure(Cliniques)etc)</li> <li>-Prise en charge de l'enrôlement de la carte Chifa avec la gestion des bordereaux</li> <li>Mandatement(Paiement des différentes prestations)</li> <li>-Gestion des états de sorties(bordereaux récapitulatifs des Tiers Payant,Ventilation des paiements)</li> <li>- Contrôle Médical (Traitement des factures conditionnées, traitement des dossiers de demandes de prise charge)</li> </ul>	01/09/2010	300G en croissance	4000

## Chapitre 4. Analyse de l'existant

---

Les Principales Applications Métiers					
Application	Utilisateurs	Fonctionnalités	Date de Mise en place	Volume	Nombres d'accès
SYSRET	Contrôle de Gestion	<ul style="list-style-type: none"> <li>- Prise en charge des dossiers de demande de pension de retraite</li> <li>-Renouvellement des pièces pour le paiement des pensions</li> <li>-Liquidation et calcul des droits</li> <li>-Révision des dossiers des pensionnés (Rappels, TP)</li> <li>- Ré-imputation (Mise à jour, réexpédition, suivi et éditions des divers états)</li> <li>- Paiement (1er paiement, Rappel, TP, ,)</li> <li>- Traitement Collectif : (Revalorisation, Relèvement du SNMG, ...)</li> </ul>	01/11/2022	100G en Croissance	1300
SIAD	Suivi décisionnel par le top management de la CASNOS et l'ensemble de ses structures décentralisées.	<ul style="list-style-type: none"> <li>-Suivi de l'activité Le recouvrement</li> <li>-Suivi de l'activité prestations fournies pour l'ensemble de ses adhérents.</li> </ul>	01/11/2022	120G en Croissance	100

## Chapitre 4. Analyse de l'existant

---

Damankoum	L'ensemble des assurés sociaux de la CASNOS	<p><b>1. Assurés Sociaux :</b></p> <ul style="list-style-type: none"> <li>-La déclaration d'activités économiques en ligne</li> <li>-La demande d'affiliation en ligne</li> <li>-La déclaration d'assiettes de cotisation annuelles</li> <li>-Le règlement des factures via le paiement électronique</li> <li>- La consultation du relevé de compte de l'assuré</li> <li>- La consultation des remboursements médicaux de l'assuré et de ses ayants-droit</li> <li>- La consultation des échéances de paiement de la pension de retraite</li> </ul> <p><b>2. Pharmaciens :</b></p> <ul style="list-style-type: none"> <li>-La consultation de la liste des médicaments remboursables et les conditions de remboursement de chaque médicament</li> <li>-La consultation de l'historique des consommations de médicament par l'assuré social et ses ayants-droit</li> <li>-Le dépôt des bordereaux pharmaceutiques sur la plate-forme</li> <li>-La consultation du traitement des bordereaux pharmaceutiques et les détails de chaque bordereau</li> <li>- Le téléchargement des fichiers de mise à jour pour l'application Pharmnos installée au leur niveau</li> <li>- La demande d'avis médical à priori pour un médicament et la réception de la réponse avec possibilité d'imprimer l'accord médical</li> </ul>	01/11/2022	75G en Croissance	3547832
-----------	---	--	------------	-------------------	---------

## 4.4 Cartographie des Applications Métiers Principales de la CASNOS

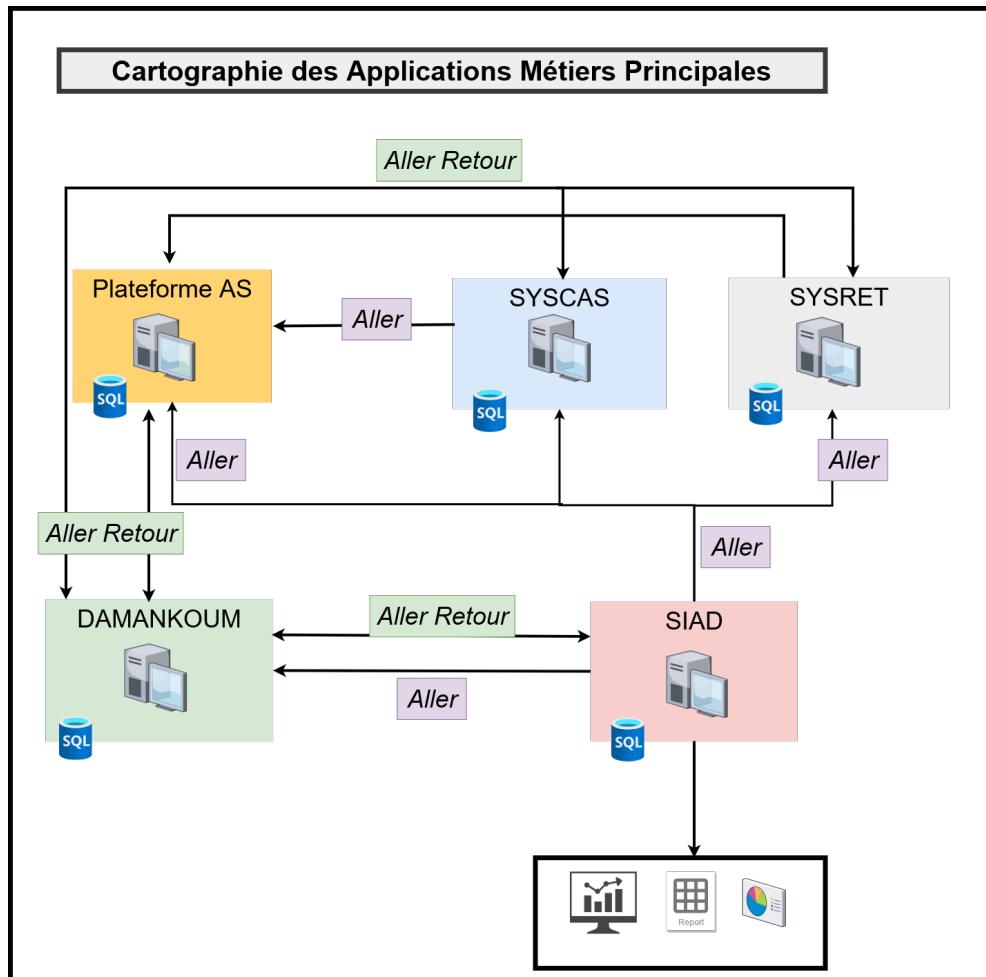


FIGURE 4.2 – Schéma des Applications Métiers Principales de la CASNOS

## 4.5 Critique de l'existant

Le but du critique de l'existant est de recenser les problèmes que trouvent les informaticiens de l'entreprise ; Dans ce cas, nous procédons à une critique objective du système en cour.

### 4.5.1 Les Problèmes trouvés dans les systèmes existants :

- Des systèmes décentralisés ce qui engendrent une difficultés dans la conception des solutions futures.
- Une forte augmentation du volume de données générées et stockées, ce qui entraîne un problème de redondance, de manipulation et surtout d'exploitation de ces données.
- La croissance du nombre de besoins d'analyse pose un problème de vélocité

- Difficultés de traitement d'un ensembles d'une certaine volumétrie
- Parfois le temps de latence est important surtout quand des transaction (Update, Insert) sont appliquées sur des tables de volumétrie importantes (37 Millions de Lignes) .
- Parfois un chargement d'une table de 5 million de lignes peut prendre jusqu'à 45 min .
- La difficulté d'accès à la données sources.
- Les tableaux de bord ne sont pas en temps réels
- Les nouveaux besoins d'analyse que la CASNOS prévoit de traiter, sont basés sur des données de types variés, ce qui impose l'utilisation d'outils spéciaux.

### **4.6 Conclusion**

Dans ce chapitre, nous avons pu comprendre et analyser la situation actuelle de la CASNOS et ses systèmes opérationnels, tout en identifiant les besoins principalement attendus dans notre projet.

Dans le prochain chapitre 'Conception Et Modélisation De La Solution' nous allons détailler et expliquer notre solution.

# Troisième partie

## Conception Et Modélisation De La Solution

# Chapitre 5

## Architectures de la plateforme

### 5.1 Architecture fonctionnelle

#### 5.2 Introduction

Après avoir fait l'analyse et l'identification des besoins présentées dans le chapitre précédent, mais aussi selon les moyens à notre disposition, le meilleur choix qui semble approprié pour une telle situation est de déployer un cluster SQL SERVER BIG DATA sur un cluster Kubernetes. Dans ce chapitre, nous allons présenter la solution proposée à savoir : l'architecture fonctionnelle, physique et technique.

### 5.3 Présentation de l'architecture fonctionnelle

Notre architecture fonctionnelle est basée sur l'architecture Lambda, nous avons opté pour ce choix car les besoins de la CASNOS nécessitent deux types de traitement, en temps réel et par lots. De plus, l'architecture Lambda présente plusieurs avantages, notamment la robustesse et la tolérance aux pannes, elle permet des traitements à faible latence, elle est évolutive, généralisée et flexible.

L'architecture fonctionnelle proposée est composée de quatre parties principales :

- **Sources** : ce sont les générateurs des données qui sera chargés et traités sur notre plateforme qui offre une variété de choix entre les BD relationnelle (SQL Server (Plateforme Assurance sociale)), Bd NOSQL ainsi que les fichiers logs avec la facilité d'intégration à notre data lake . Cette collecte se fera pour chaque pipeline (temps réel et batch) où on utilisera le connecteur SQL Server JDBC qui servira à récupérer les données existantes.

- **Ingestion** : ingestion de données sur notre plateforme est assurée dans cette partie, en utilisant des outils permettant le transfert des données depuis les sources jusqu'au services de stockage ou de traitement. Nous pouvons citer SQL SERVER INTÉGRATION SERVICE pour l'importation des données SQL SERVER , ou encore la Data Virtulisation (Poly Base) qui assure l'intégration de divers type de données tel ( MongoDB, ORACLE)
- **Digestion** : Cette partie est le centre de notre plateforme, c'est là où les données vont

être stockées, traitées et transformées en valeurs à consommer par les utilisateurs finaux.

- **Les cas d'usage :** Le but initiale de ce PFE est de mettre en place une plateforme SQL SERVER BIG DATA CLUSTER, mais nous avons pensé à quelques cas d'usage afin de prouver le bon fonctionnement de la plateforme. A titre d'exemple : Le monitoring réel time avec Graphana, machine learning etc.

La figure ci-dessous illustre notre proposition d'architecture fonctionnelle .

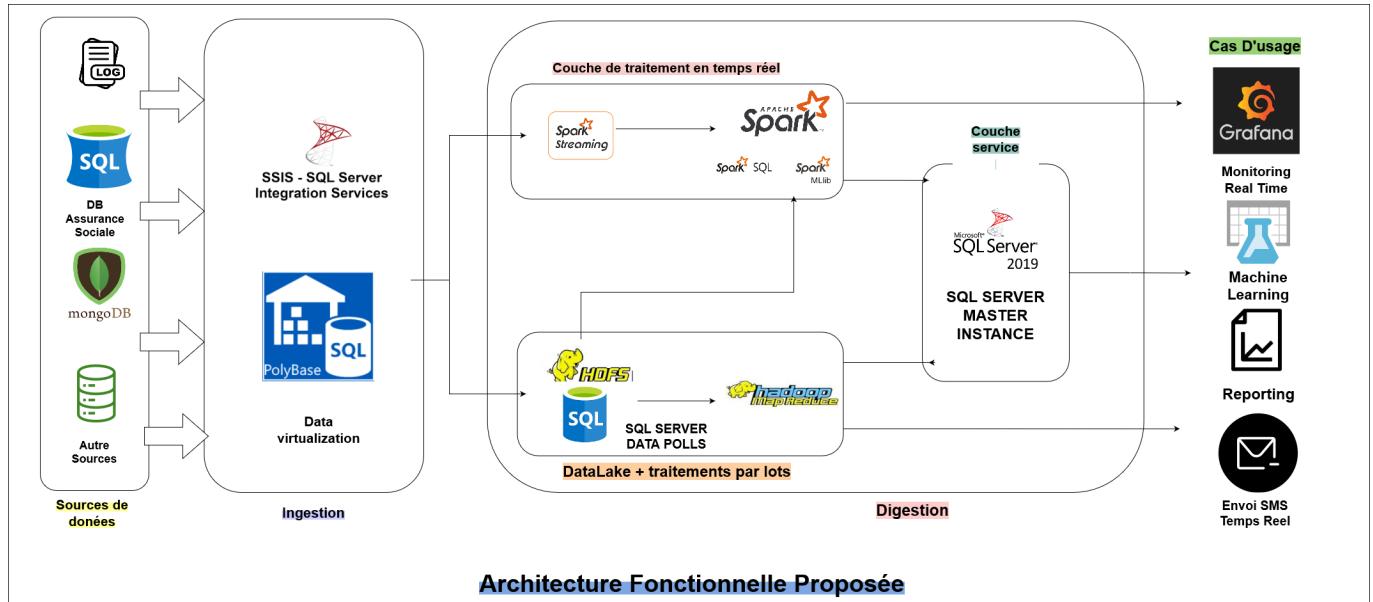


FIGURE 5.1 – Architecture fonctionnelle proposée

## 5.4 Architecture Technique

### 5.4.1 Qu'est ce qu'une architecture Technique ?

L'architecture technique est une vue tournée sur l'organisation logique de la plateforme informatique, c'est-à-dire les moyens techniques clés qui seront utilisés par tous les logiciels applicatifs.

Les spécifications techniques décrivent l'implémentation du programme. Quelle va être la base de données ? Son organisation ? Quel langage va-t-on choisir, quel framework ? L'objectif de ce document est de préparer en amont tous les éléments techniques dont les développeurs auront besoin pour coder.

Autrement dit : l'architecture technique = Comment repartir l'architecture fonctionnelle sur un ensemble de machine. Le digramme qui suit illustre notre architecture technique .

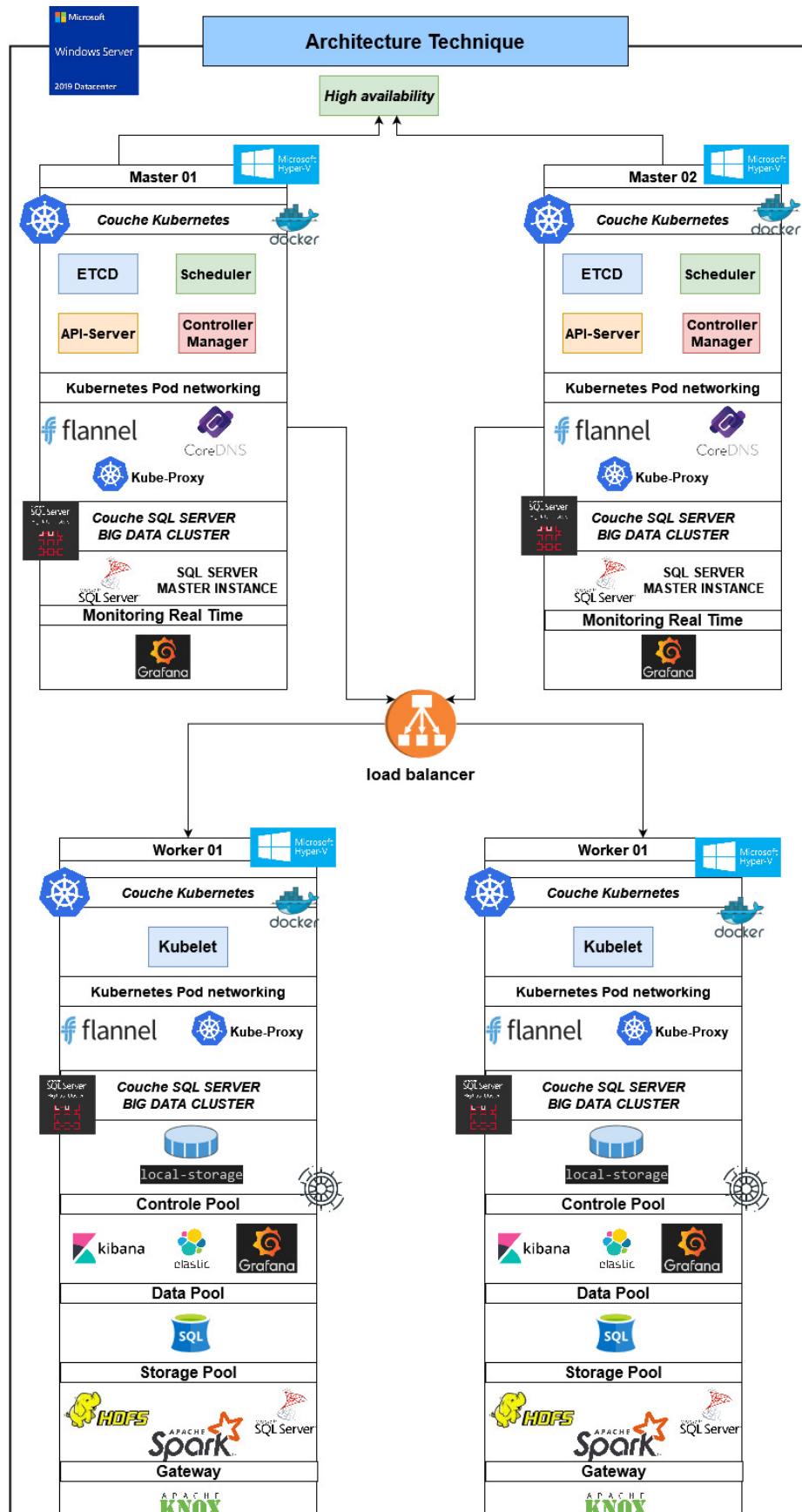


FIGURE 5.2 – Architecture Technique Proposée

### 5.4.2 Justification de l'architecture technique

• Faire un mélange entre VM et Conteneurs DOCKER afin d'avoir les 3 concepts : la virtualisation, la conteneurisation et microservices. Afin d'assurer :

1. L'évolutivité : L'adaptation au changement
2. Scalabilité : S'adapter aux fluctuations de la demande en conservant ses différentes fonctionnalités.
3. Haute Disponibilité
4. Maintenabilité : Capacité d'un système à être simplement et rapidement réparé
5. Accessibilité + sécurité : Un produit mise à la disposition d'un grand nombre .

### 5.4.3 Pourquoi cette séparation virtualisée ?

Nous ne pouvons pas construire tout le système sur un seul Docker engine car en cas de pannes on risque de tout perdre (maillon faible).

### 5.4.4 Décortiquer l'architecture technique

En génie logiciel, un cluster ressemble à un groupe de nœuds qui travaillent ensemble pour répartir la charge de travail. De plus, le clustering contribue à la tolérance aux pannes, en ayant un cluster agissant comme secondaire (sauvegarde) par rapport à un cluster principal. En gros, l'architecture se devise en 2 types de noeuds : **Noeuds Master +Noeuds Worker**.

#### 5.4.4.1 Noeuds Master

Ce sont les noeuds maîtres qui gèrent le Clsuter Kubernetes . Un nœud maître est un nœud qui contrôle et gère un ensemble de nœuds de travail (exécution des charges de travail) et ressemble à un cluster dans Kubernetes. Un nœud maître possède les composants suivants pour faciliter la gestion des nœuds de travail :

##### — Couche Kubernetes :

1. **Kube-APIServer** : Qui agit comme l'interface du cluster. Toutes les communications externes au cluster se font via le serveur API.
2. **Kube-Controller-Manager** : Le contrôleur-gestionnaire met en œuvre la gouvernance dans l'ensemble du cluster.
3. **Etcd** : La base de données d'état du cluster.
4. **Kube Scheduler** : Qui planifie les activités des nœuds de travail en fonction des événements se produisant sur etcd.

— **Kubernetes Pod Networking** Consiste la couche réseau kubernetes, elle se compose de :

1. **Flannel(CNI<sup>1</sup>)** est une couche simple pour Kubernetes. Flannel gère un Réseau IPv4 entre plusieurs noeuds dans un cluster. Il ne contrôle pas la façon dont les conteneurs sont mis en réseau avec l'hôte, uniquement la façon dont le trafic est transporté entre les hôtes.
2. **CoreDNS** : CoreDNS est un DNS<sup>2</sup>. Il est écrit en Go. Il peut être utilisé dans une multitude d'environnements en raison de sa flexibilité. CoreDNS est sous licence Apache, et entièrement open source.
3. **Kube-Proxy** : Kube-proxy est un proxy réseau qui s'exécute sur chaque noeud dans le cluster, mise en œuvre d'une partie de Kubernetes de services. kube-proxy maintient les règles du réseau sur les noeuds. Ces règles de réseau permettent au réseau communication avec les pods à partir de sessions réseau à l'intérieur ou à l'extérieur du cluster.

— **Couche SQL SERVER BIG DATA CLUSTER**

1. **L'instance maître SQL** : The master pool contient l'instance maître de SQL Server. L'instance maître gère la connectivité (Azure Data Studio ou SQL Server Management Studio), les requêtes scale-out, les métadonnées (HDFS, tables externes), les bases de données utilisateur et les services d'apprentissage automatique.
2. **Monitoring Real-Time Via Graphana** : L'un des avantages de la plate-forme, c'est la possibilité de visualisation des états de nodes, transaction sql server ainsi que l'état des composants du sql server big data.

**Note :** Les éléments qui existent sur le Master 01 sont les même du Master 02 , cela assure la haute disponibilité : si le master 01 tombe en panne le master 02 va le remplacer sans cause de problèmes en production .

---

1. Container Network Interface (CNI) est un framework de configuration dynamique des ressources réseau. Il utilise un ensemble de bibliothèques et de spécifications écrites en Go. La spécification du plugin définit une interface pour configurer le réseau, fournir des adresses IP et maintenir la connectivité avec plusieurs hôtes.

2. Le Domain Name System ou DNS est un service informatique distribué utilisé qui traduit les noms de domaine Internet en adresse IP ou autres enregistrements.

### 5.4.4.2 Noeuds Workers

- **Couche Kubernetes : Kubelet** : Kubelet, d'autre part, est un processus qui s'exécute sur chaque nœud d'un cluster Kubernetes. Il peut créer, détruire ou mettre à jour des pods et leurs conteneurs Docker pour le nœud donné lorsqu'il est invité à le faire.
- **Kubernetes Pod Networking** : Même que Master Node.
- **Couche SQL SERVER BIG DATA CLUSTER**
  1. **Local-Storage** Ce sont les volumes persistants fournissent un modèle de plugin pour le stockage dans Kubernetes. C'est l'endroit où tout les fichiers reler à sql server bdc sont stockés. Afin d'assurer un bon contrôle sur le type de stockage, la disponibilité et les performances. Kubernetes fait en sorte de créer deux copies (Worker01+Worker02).
  2. **Contrôle Pool** : Se compose de diverses technologies permettant d'avoir une vue globale sur le cluster comme **Graphana**. En outre, SQL SERVER BDC nous donne la possibilité de stocker/extrait les fichiers logs ainsi d'en sortir des dashboards pour la visualisation real-time (Kibana +ElasticSearch)
  3. **Data Pool** : Citons quelques scénarios de pool de données : une requête complexe joignant plusieurs sources de données PolyBase (Logs MongoDB), utilisée pour un rapport hebdomadaire, pourrait être déchargée dans le pool de données. De même, les données du tableau de bord (Assurance Sociale) nécessitant une actualisation périodique peuvent être mises en cache dans le pool de données pour des rapports optimisés.
  4. **Storage Pool** : Les nœuds de stockage sont responsables de : Ingestion de données via Apache Spark. Utilisation de tables externes SQL Server pour permettre l'interrogation des données à l'aide des nœuds de calcul PolyBase et des instances SQL Server s'exécutant dans les nœuds HDFS.
  5. **Gateway** : Passerelle pour accéder aux fichiers HDFS, Spark (Knox) - Point de terminaison HTTPS pour accéder à des services tels que webHDFS et Spark.

**Note :** Les éléments qui existent sur le worker01 sont les mêmes du worker02, cela assure la haute disponibilité ainsi que la répartition de charge.

## 5.5 Architecture Physique

Les configurations matérielle et logicielle qui nous ont été attribuées se composent d'un serveur : *Fujitsu Primergy TX200 s6* à 64G RAM et 800G de stockage et un réseau Ethernet qui lie le serveur au réseau de la CASNOS. Le système d'exploitation utilisé sur chaque serveur est **Windows Server 2019 DATACENTER**, nous avons pu créer les machines virtuelles à l'aide de l'outil **Hyper-V** fourni par la société, le système d'exploitation utilisé sur chaque machine virtuelle est **Ubuntu 20.04**.

Ci dessous un résumé de la configurations matérielle.

Processeur	RAM	Stockage	Réseau	OS	Outils
intel xeon x5670 processor 2.93 ghz	64GO	800GO	Local + Internet	Windows Server 2019 DATACENTER + Ubuntu 20.04(VMs)	Hyper-V

### 5.5.1 Explication de l'architecture physique

Le serveur est reliée à un switch DELL permettant de relier nos laptops afin d'avoir la possibilité d'utilisation de bureau à distance (avec câble). La liaison réseau des machines physiques attribuées permettre aux instances de notre plateforme de communiquer entre elles tout en autorisant l'accès Internet afin d'effectuer les installations et les mises à jour logicielles nécessaires. L'image qui suit représente l'architecture physique de la Plateforme.  
**Note :** Dans l'architecture technique, il était prévue de créer 5VMs : 2Masters +2 Workers +load balancer (répartition de charge) pour assurer la haute disponibilité, ce qui ne sera pas possible car les Vms Kubernetes sont gourmandes en matière de ressources. En outre les exigences minimales/noeuds pour le déploiement de SQL SERVER BIG DATA CLSUETR sont : **8 CPU, 64 GB RAM , 100 GB disk space**

- Donc à défaut de ressources nous avons réaliser un cluster avec le minimum de Vms : en créant 3 Vms ubuntu 20.04 : 1 Master + 2 Workers avec les caractéristiques suivantes :
  - Master : 13G RAM + 170 G Stockage
  - Worker 1 : 25G RAM + 230G Stockage
  - Worker 2 : 10G RAM + 230G Stockage
- **Calcule de répartition de la RAM :**  $13(\text{Master}1)+25(\text{Worker}1)+10(\text{Worker}02)=48$   
 $64-48= 16\text{G}$  (Pour le Système OS , Windows server 2019)

### 5.5.2 Exigences minimales des technologies :

- **Kubernetes** : 8 G Memory /8 CPU / 100 G STORAGE
- **SSIS** : 6GB Memory / 4 Core /30 GB
- **SQL SERVER BIG DATA CLUSTER** : 8Memory /8 CORE / 40 GB

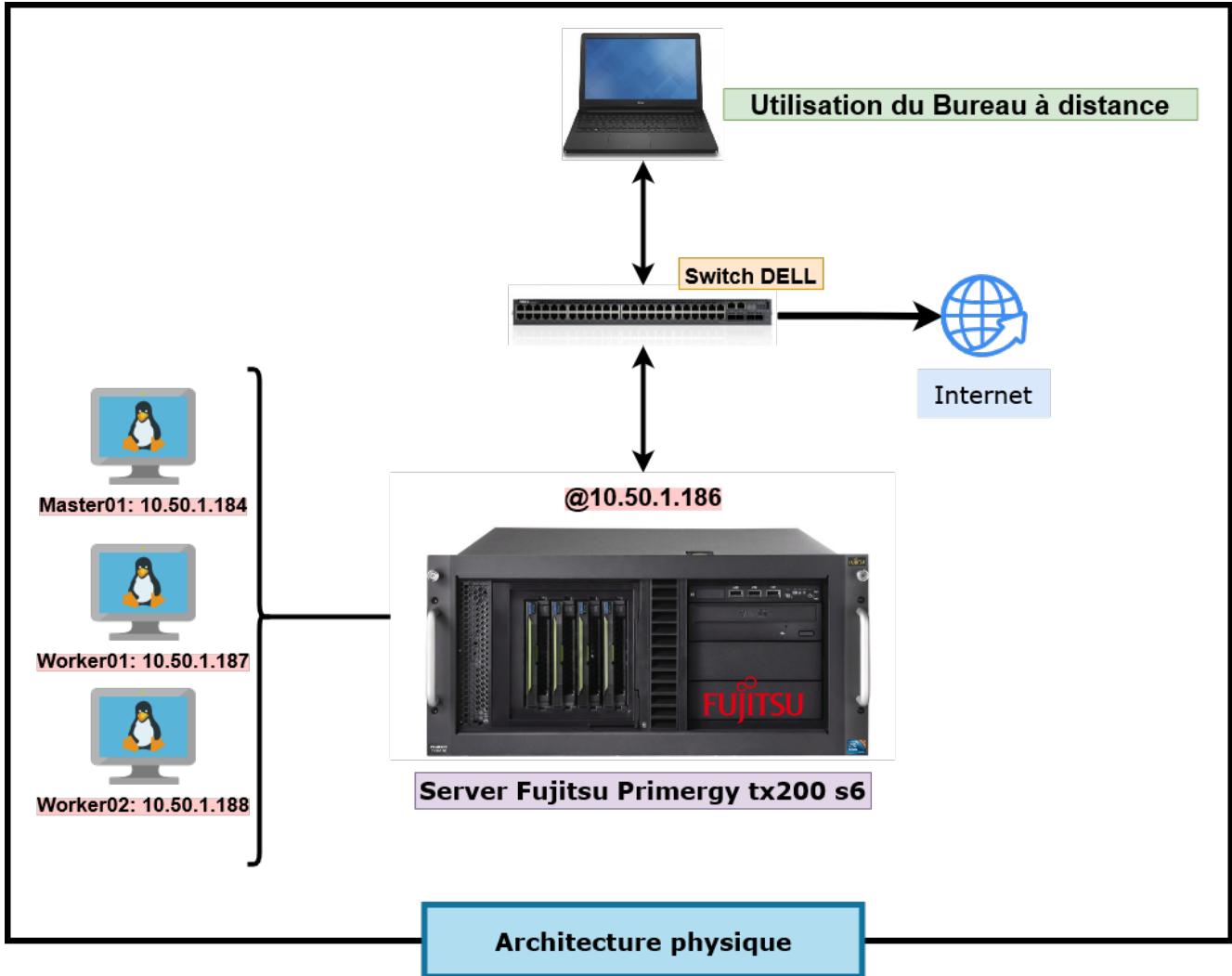


FIGURE 5.3 – Architecture Physique Proposée

## 5.6 Structure en couches de la plateforme

Le diagramme ci dessous liste les différentes couches intégrées de notre plateforme. Nous pouvons les énumérer comme suit :

1. HARDWARE Pour Plus De Détailles consulter l'ANNEXE A
2. HYPER-V Pour Plus De Détailles consulter l'ANNEXE A
3. Kubernetes Cluster
4. SQL SERVER BIG DATA CLUSTER
5. Services : Cas d'usages

**Note** :Consulter l'ANNEXE B afin d'avoir une idée sur les raisons du choix technologique.

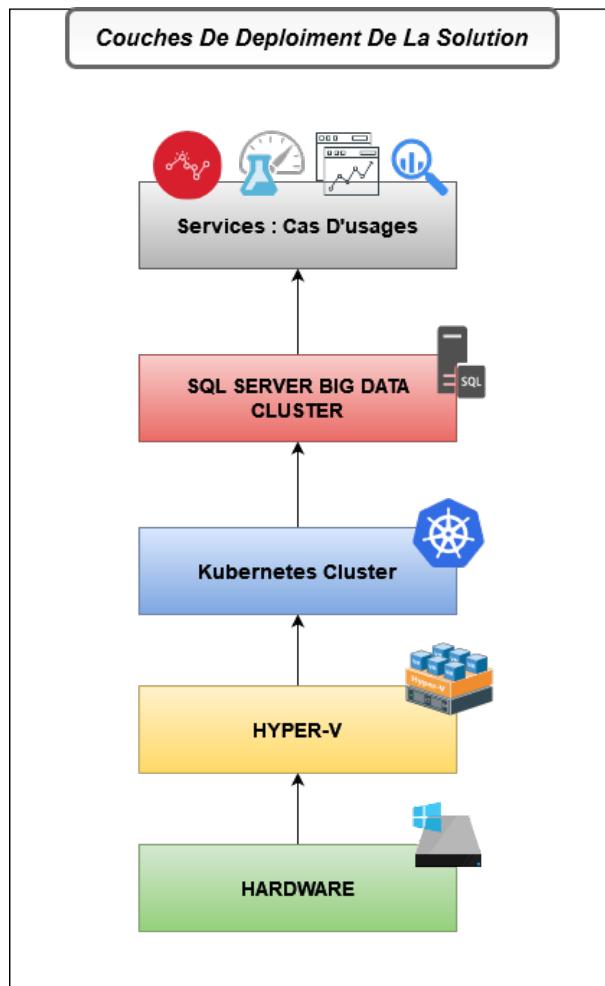


FIGURE 5.4 – Couches de la plateforme

### 5.6.1 Hardware

De manière générale, le matériel informatique est une partie ou un composant d'un système informatique. C'est la partie physique de l'ordinateur qui s'interface avec le logiciel. À l'intérieur de l'appareil, il y a des composants essentiels à son fonctionnement et d'autres composants secondaires qui sont à l'extérieur. Dans notre cas, il s'agit des machines physiques ou Serveurs.

### 5.6.2 Hyper-V

C'est la couche de visualisation, Hyper-v permet la création des machines virtuelles tout en prenant en compte le minimum de ressources pour chaque machine.

### 5.6.3 Kubernetes Cluster

Représente la partie de conteneurisation et orchestration de conteneurs. Kubernetes joue le rôle de chefs d'orchestre et les conteneurs d'SQL SERVER BIG DATA CLUSTER vont jouer le rôle des musiciens (Gérés par kubernetes).

#### **5.6.4 SQL SERVER BIG DATA CLUSTER**

Apres avoir préparer les couches précédentes, nous pouvons maintenant déployer un cluster SQL SERVRE BIG DATA CLUSTER sur un cluster Kubernetes afin d'assurer un bon fonctionnement.

#### **5.6.5 Services : Cas d'usages**

Après le déploiement de la plate forme , viens l'étape de son exploitation via des cas d'usage tel :lancement des job Apache Spark, Machine Learning, Monitoring etc.

### **5.7 Conclusion**

Dans ce chapitre, nous avons présenté les architectures de notre plateforme, y compris l'architecture fonctionnelle, physique et technique. En nous basant sur ces architectures, nous essayerons dans ce qui suit de proposer la solution la plus adéquate possible pour répondre aux besoins.

# Quatrième partie

## Déploiement de la solution

# Chapitre 6

## Déploiement du cluster kubernetes

### 6.1 Introduction

Dans ce chapitre, nous allons présenter en détails la phase de déploiement d'un cluster Kubernetes basée sur Kubeadm localement sur des serveurs de la CASNOS.

### 6.2 Organisation des fichiers de commandes

Afin d'assurer une bonne organisation, nous avons créer des fichiers d'exécution en rassemblant les commandes qui permettent le déploiement d'un cluster Kubernetes en utilisant Kubeadm. D'abord nous avons crée un dossier "K8S" pour mettre les fichier de configuration dedans. Nous avons utiliser MobaXterm SSH afin de facilite la navigation et l'utilisation du terminale.

1. **Setup-k8s-prereqs.sh** : Permet d'installer les composant important pour préparer les machines afin de supporter le cluster.
2. **Setup-k8s-master.sh** : Permet d'initialiser le cluster
3. **Pre-Join-Worker** : Sert à préparer les workers (Esclaves) pour joindre le Cluster.

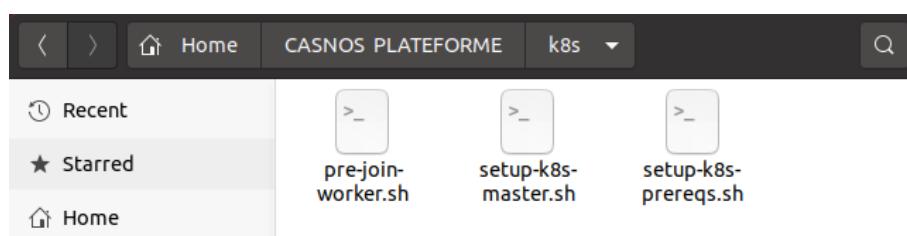


FIGURE 6.1 – Organisation Des Fichiers De configuration

## 6.3 Pré-requis pour la configuration de Kubeadm

- Au moins deux noeuds Ubuntu [un maître et un nœud de travail].
- Le nœud maître doit avoir au minimum 2 vCPU et 2 Go de RAM .
- Pour les noeuds worker, un minimum de 1 vCPU et 2 Go de RAM est recommandé.
- Connectivité réseau entre toutes les machines du cluster, qu'il soit public ou privé.

## 6.4 Préparation des machines pour supporter le cluster Kubernetes

Le fichier "Setup-k8s-prereqs.sh" regroupe toutes les commandes qui doivent être exécuter afin de préparer les machine.

Nous avons regrouper les commandes qui doivent être exécutées avant de lancer le cluster. Il consiste à la configuration des VMs, installation de l'exécution du conteneur(Nous utiliserons Docker). Puis viens l'installation des composant K8S : Kubeadm, Kubelet et kubectl sur tous les nœuds.

Voici un aperçu sur le fichier .

```
#!/bin/sh -x #Exécuter des Fichiers Shell(CMD)

# Setup the kubernetes prerequisites
sudo apt install curl
echo $(hostname -i) $(hostname) >> /etc/hosts
sudo sed -i "/swap/s/^/#/" /etc/fstab
sudo swapoff -a } Préparation Des VMs

apt-get install -y docker.io } Installation De Docker

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
KUBE_DPKG_VERSION=1.24.0-00
apt-get update
apt-get install -y apt-transport-https
apt-get install -y kubelet=$KUBE_DPKG_VERSION kubeadm=$KUBE_DPKG_VERSION
kubectl=$KUBE_DPKG_VERSION
curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get | bash } Installation Des Composant
Kubernetes : Kubeadm , Kubelet, Kubectl

apt-get install -y ebttables ethtool
. /etc/os-release
if [ "$UBUNTU_CODENAME" = "bionic" ]; then
    modprobe br_netfilter
fi
sysctl net.bridge.bridge-nf-call-iptables=1 } Configurer Le réseau
Entre Les Vms
```

FIGURE 6.2 – Setup-k8s-prereqs.sh

## 6.5 Initialiser le cluster sur le noeud maître à l'aide de Kubeadm

Afin d'initialiser Kubeadm sur le noeud maître, il faut suivre les étapes du fichier **Setup-k8s-master.sh** :

1. Rafraîchir le noeud : En relancer les composant initiale : docker, containerd, et les éléments de kubernetes.
2. Initialiser les configurations du plan de contrôle du noeud maître à l'aide de la commande kubeadm suivante.

```
Kubernetes Version  
=1.24.0  
  
KUBE_VERSION=1.24.0  
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --kubernetes-version=$KUBE_VERSION --v=5  
  
Plage D'adresses IP  
Pour Les Pods  
  
Avoir Plus de Détails  
lors de L'initialisation
```

FIGURE 6.3 – Kubeadm Init

3. Lors d'une initialisation réussie de kubeadm, nous obtenons une sortie avec l'emplacement du fichier kubeconfig, avec un jeton de connexion.

```
Your Kubernetes control-plane has initialized successfully!  
  
To start using your cluster, you need to run the following as a regular user:  
  
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

FIGURE 6.4 – Emplacement Du Fichier KubeConfig

4. Ensuite, installer le plug-in de réseau(CNI)<sup>1</sup> Flannel pour la mise en réseau des pods .



FIGURE 6.5 – Plug-IN Réseau Flannel

1. Container Network Interface (CNI) est un cadre de configuration dynamique des ressources réseau. Il utilise un ensemble de bibliothèques et de spécifications écrites en Go. La spécification du plug-in définit une interface pour configurer le réseau, fournir des adresses IP et maintenir la connectivité avec plusieurs hôtes

Voici une vue globale du fichier Setup-k8s-master.sh

```

#!/bin/sh -x
# Initialize a kubernetes cluster on the current node.

systemctl restart containerd
systemctl restart kubelet
| ufw disable
swapoff -a; sed -i '/swap/d' /etc/fstab
sudo kubeadm reset
sudo rm -rf $HOME/.kube
} } 
```

Rafraîchir le noeud (Restart Componant)

```

KUBE_VERSION=1.24.0
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --kubernetes-version=$KUBE_VERSION --v=5
} } 
```

Initialiser Le Cluster Avec Kubeadm

```

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
} } 
```

Obtention de l'emplacement du fichier "KUBECONFIG"

```

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master
/Documentation/kube-flannel.yaml
kubectl apply -f rbac.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy
/recommended/kubernetes-dashboard.yaml
kubectl create clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin
--serviceaccount=kube-system:kubernetes-dashboard
} } 
```

1) Installation du CNI Flannel pour la mise en réseau des POD  
2) Kubernetes Tableau de bord pour la visualisation du Cluster

FIGURE 6.6 – Setup-k8s-master.sh

## 6.6 Joindre les noeuds workers

Nous avons également configuré les utilitaires Docker, kubelet et kubeadm sur les noeuds de travail. Maintenant, joignons le noeud de travail au noeud maître à l'aide de la commande Kubeadm join que nous avons dans la sortie lors de la configuration du noeud maître.

Si vous avez manqué la commande join, exécutez la commande suivante dans le noeud maître pour recréer le jeton.

```

root@srv-k8s-master-01:/home/srv-k8s-master-01#kubeadm token create --print-join-command
kubeadm join 10.50.1.184:6443 --token p8vuph.zhk5uch03wcik510 \ --discovery-token-ca-cert-hash
sha256:102519d6473a201fc3442981719fd78547e127a57387ffeb506c6def240816ef
} 
```

FIGURE 6.7 – Kubeadm Token Join

Afin de joindre les noeuds au cluster il faut exécuter la sortie de la commande précédente dans le terminal du worker node. **Note :** Vu que dans notre architecture, nous avons 2 Vms Worker donc nous exécuterons aussi cette commande dans "srv-k8s-worker-02". La figure ci dessous montre que le noeud est bien ajouté au maître.

```
root@srv-k8s-worker-01:/home/srv-k8s-worker-01/k8s# kubeadm join 10.50.1.184:6443 --token p8vuph.zhk5uch03wcik510 --discovery-token-ca-cert-hash sha256:102519d6473a201fc3442981719fd78547e127a57387ffeb506c6def240816ef
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

FIGURE 6.8 – Joindre Le Noeud Worker

## 6.7 Quelques commandes kubernetes

### 6.7.1 Vérifier l'état du cluster

En utilisant la commande "*kubectl get nodes*" qui permet d'afficher tout les éléments du cluster, leurs rôles, état (READY /NOT READY), version etc.

```
root@srv-k8s-master-01:/home/srv-k8s-master-01/k8s# kubectl get nodes
NAME           STATUS    ROLES      AGE     VERSION
srv-k8s-master-01   Ready    control-plane   3m2s   v1.24.0
srv-k8s-worker-01   Ready    worker       82s    v1.24.0
srv-k8s-worker-02   Ready    worker       75s    v1.24.0
```

FIGURE 6.9 – Vérifier L'état Du Cluster

Comme illustre l'image ci-dessus, nous voyons :

1. Les 3 noeuds du cluster : svr-k8s-master-01 , svr-k8s-worker-01, svr-k8s-worker-02
2. L'état : le statut des noeuds est ready (Prêt).
3. Le rôle de chaque noeuds : controle-plane (MASTER) et Worker

### 6.7.2 Lister les pods du cluster

En utilisant la commande `kubectl get pods -A` qui permet de lister tout les pods du cluster, ainsi que le name-space<sup>2</sup> auquel ils appartient, leurs états , le statut et âge.

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-flannel	kube-flannel-ds-h4rk9	1/1	Running	0	3m52s
kube-flannel	kube-flannel-ds-nvfd5	1/1	Running	0	3m45s
kube-flannel	kube-flannel-ds-zzp2n	1/1	Running	0	4m49s
kube-system	coredns-6d4b75cb6d-dddn	1/1	Running	0	5m3s
kube-system	coredns-6d4b75cb6d-mj9mn	1/1	Running	0	5m3s
kube-system	etcd-srv-k8s-master-01	1/1	Running	0	5m26s
kube-system	kube-apiserver-srv-k8s-master-01	1/1	Running	0	5m30s
kube-system	kube-controller-manager-srv-k8s-master-01	1/1	Running	34	5m29s
kube-system	kube-proxy-l8swp	1/1	Running	0	5m4s
kube-system	kube-proxy-mtgqn	1/1	Running	0	3m45s
kube-system	kube-proxy-sfhrb	1/1	Running	0	3m52s
kube-system	kube-scheduler-srv-k8s-master-01	1/1	Running	34	5m26s
kube-system	kubernetes-dashboard-cd4778d69-kcg7s	1/1	Running	0	

FIGURE 6.10 – Lister Les Pods Du Cluster

## 6.8 Conclusion

À ce stade, nous devrions disposer d'un cluster kubernetes entièrement fonctionnel sur lequel nous pouvons exécuter des services et des charges de travail. Si vous vous demandez quoi faire avec le cluster maintenant qu'il est configuré, une bonne prochaine étape serait de déployer le SQL SERVER BIG DATA CLUSTER.

---

2. Ce sont un moyen d'organiser les clusters en sous-clusters virtuels. Ils peuvent être utiles lorsque différentes équipes ou projets partagent un cluster Kubernetes. N'importe quel nombre d'espaces de noms est pris en charge dans un cluster, chacun étant logiquement séparé des autres mais avec la possibilité de communiquer entre eux.

# Chapitre 7

## Déploiement du big data cluster

### 7.1 Introduction

Après avoir préparer la couche Hyper-V et maintenant que le cluster Kubernetes fonctionne et que tous les nœuds de travail sont joints, il est temps de lancer le déploiement du cluster Big Data. Dans ce chapitre nous allons voir toutes les phases en détails .

### 7.2 Étapes du déploiement

Dans cette section nous allons lister toutes les étapes à suivre afin d'avoir un cluster SQL SERVER BIG DATA sous K8s.

1. Commençons par l'installation de *AZDATA(Azure Data CLI)* via la commande suivante.(Voir Figure ci-dessous )

```
sudo apt-get update  
sudo apt-get install -y azdata-cli
```

FIGURE 7.1 – Installation Azure Data CLI

2. SSH sur le nœud maître et exécuter la commande azdata ci-dessous qui permet de lancer la création du BDC.

```
azdata bdc create  
The privacy statement can be viewed at:  
https://go.microsoft.com/fwlink/?LinkId=853010  
  
The license terms for SQL Server Big Data Cluster can be viewed at:  
Enterprise: https://go.microsoft.com/fwlink/?LinkId=2104292  
Standard: https://go.microsoft.com/fwlink/?LinkId=2104294  
Developer: https://go.microsoft.com/fwlink/?LinkId=2104079  
  
Do you accept the license terms? (y/n): y
```

FIGURE 7.2 – Lancement de la création du BDC

3. Accepter les termes en appuyant sur y puis sur retour.
4. Pour le type de déploiement, choisir l'option 5 (« kubeadm-dev-test ») car nous avons utiliser kubeadm et c'est un environnement de test et non de production puis appuyer sur retour.

```
Please choose a deployment configuration:  
To see more info please exit create and use command [bdc config list -c <config_profile>]  
  
1. aks-dev-test  
2. aks-dev-test-ha (default choice)  
3. aro-dev-test  
4. aro-dev-test-ha  
5. kubeadm-dev-test  
6. kubeadm-prod  
7. openshift-dev-test  
8. openshift-prod  
  
Enter your choice as a number or unique substring (Control-C aborts): 5
```

FIGURE 7.3 – Choisir le type de déploiement du BDC

5. Taper le nom d'utilisateur/mot de passe azdata

```
Enter your choice as a number or unique substring (Control-C aborts): 5  
  
Azdata username:casnos  
Azdata password:  
Confirm Azdata password:
```

FIGURE 7.4 – Compte azdata du BDC

6. Taper « local-storage » pour de classe de stockage kubernetes (deux fois).

```
Needed Configuration Values in control.json  
  
Kubernetes Storage Class  
- Config Path: spec.storage.data.className  
- Description: This indicates the name of the Kubernetes Storage Class to use. You must pre-provision the storage class and the persistent volumes or you can use a built in storage class if the platform you are deploying provides this capability.  
- Please provide a value: local-storage
```

FIGURE 7.5 – Préciser la classe de stockage du BDC

7. Nous pouvons voir "Démarrage du déploiement du cluster" comme indiqué dans la capture dans la page suivante qui illustre une vue globale de la commande *Azdata BDC Create*.

```
root@srv-k8s-master-01:/home/srv-k8s-master-01# azdata bdc create
The privacy statement can be viewed at:
https://go.microsoft.com/fwlink/?LinkId=853010

The license terms for SQL Server Big Data Cluster can be viewed at:
Enterprise: https://go.microsoft.com/fwlink/?linkid=2104292
Standard: https://go.microsoft.com/fwlink/?linkid=2104294
Developer: https://go.microsoft.com/fwlink/?linkid=2104079

Do you accept the license terms? (y/n): y

Cluster deployment documentation can be viewed at:
https://aka.ms/bdc-deploy

Please choose a deployment configuration:
To see more info please exit create and use command [bdc config list -c <config_profile>]

1. aks-dev-test
2. aks-dev-test-ha (default choice)
3. aro-dev-test
4. aro-dev-test-ha
5. kubeadm-dev-test
6. kubeadm-prod
7. openshift-dev-test
8. openshift-prod

Enter your choice as a number or unique substring (Control-C aborts): 5

Azdata username:casnos
Azdata password:
Confirm Azdata password:

Needed Configuration Values in control.json

Kubernetes Storage Class
- Config Path: spec.storage.data.className
- Description: This indicates the name of the Kubernetes Storage Class to use. You must pre-provision the storage class and the persistent volumes or you can use a built in storage class if the platform you are deploying provides this capability.
- Please provide a value: local-storage

Kubernetes Storage Class
- Config Path: spec.storage.logs.className
- Description: This indicates the name of the Kubernetes Storage Class to use. You must pre-provision the storage class and the persistent volumes or you can use a built in storage class if the platform you are deploying provides this capability.
- Please provide a value: local-storage
NOTE: Cluster creation can take a significant amount of time depending on configuration, network speed, and the number of nodes in the cluster.

Starting cluster deployment.
```

FIGURE 7.6 – Création du BDC

8. Après un peu de temps (De 45min à 1H) , le cluster sera enfin créé avec tout ses éléments (cluster control plane ,data pool ,storage pool,compute pool et master pool) .Voir figure -page suivante-

```
Starting cluster deployment.
Cluster controller endpoint is available at 10.50.1.187:30080.
Waiting for control plane to be ready after 5 minutes.
Cluster control plane is ready.
Data pool is ready.
Storage pool is ready.
Cluster 'mssql-cluster' is not ready after 15.0 minutes. Check controller logs for more details.
Compute pool is ready.
Master pool is ready.
Cluster 'mssql-cluster' deployed successfully.
```

FIGURE 7.7 – SQL SERVER BIG DATA CLUSTER Déployé

### 7.3 Connexion au cluster

#### 7.3.1 Connexion à l'aide de AZDATA

Nous devons nous connecter à azdata en faisant une connexion azdata puis en lui donnant l'espace de noms : mssql-cluster, puis nom d'utilisateur et mot de passe. Cela définira le cluster mssql comme espace de noms par défaut.

```
root@srv-k8s-master-01:/home/srv-k8s-master-01# azdata login
Namespace: mssql-cluster
Username: casnos|
Password:
Logged in successfully to `https://10.50.1.188:30080` in namespace `mssql-cluster`. Setting active context to `mssql-cluster`.
```

FIGURE 7.8 – Connexion Au Cluster

#### 7.3.2 Afficher les endpoints du cluster

Ensuite, exécuter la commande ci-dessous pour obtenir une liste de tous les points de terminaison *endpoints*.<sup>1</sup>

##### Note :

Nous pouvons désormais utiliser SQL Server Master Instance Frontend Endpoint pour se connecter via Azure Data Studio ou SSMS ou bien VS CODE .

---

1. Un endpoint est un terminal qui peut être connecté à un réseau comprenant des ordinateurs de bureau, des ordinateurs portables, des téléphones mobiles, des tablettes et des serveurs. Dès qu'un appareil est connecté à un réseau, on peut le considérer comme un endpoint.

## Chapitre 7. Déploiement du big data cluster

Description Protocol	Endpoint	Name
Gateway to access HDFS files, Spark https	https://10.50.1.188:30443	gateway
Spark Jobs Management and Monitoring Dashboard https	https://10.50.1.188:30443/gateway/default/sparkhistory	spark-history
Spark Diagnostics and Monitoring Dashboard https	https://10.50.1.188:30443/gateway/default/yarn	yarn-ui
Application Proxy https	https://10.50.1.188:30778	app-proxy
Management Proxy https	https://10.50.1.187:30777	mngmtproxy
Log Search Dashboard https	https://10.50.1.187:30777/kibana	logsui
Metrics Dashboard https	https://10.50.1.187:30777/grafana	metricsui
Cluster Management Service https	https://10.50.1.188:30080	controller
SQL Server Master Instance Front-End	10.50.1.188,31433	sql-server-master tds
HDFS File System Proxy https	https://10.50.1.188:30443/gateway/default/webhdfs/v1	webhdfs
Proxy for running Spark statements, jobs, applications	https://10.50.1.188:30443/gateway/default/livy/v1	livy
Hadoop KMS proxy for managing Hadoop Encryption keys	https://10.50.1.188:30443/gateway/default/hadoopkms/v1	hadoopkms

FIGURE 7.9 – Endpoints du cluster

## 7.4 Connecter le Cluster à AZURE DATA STUDIO

### 7.4.1 AZURE DATA STUDIO, c'est quoi ?

Conçu pour se concentrer sur les fonctionnalités de la plateforme de données que les développeurs utilisent le plus, Azure Data Studio offre des expériences supplémentaires disponibles sous forme d'extensions optionnelles. Il est conçu pour les professionnels des données qui utilisent les bases de données SQL Server et Azure sur site ou dans des environnements multiclouds.

### 7.4.2 Se connecter au cluster

Ici , nous allons utiliser une connexion "SQL LOGIN". Il faut Remplir les champs comme suit(Voir Figure page-suivante)

- **Nom du serveur :** Nous allons utiliser l'endpoint de l'instance master :10.50.1.187,31433
- **Type d'authentification :** SQL LOGIN
- **Nom d'utilisateur :** "casnos"
- **Mot de passe :** mot de passe pour le serveur SQL.
- **Nom de la base de données :** <Par défaut>
- **Groupe de serveurs :** <Par défaut>

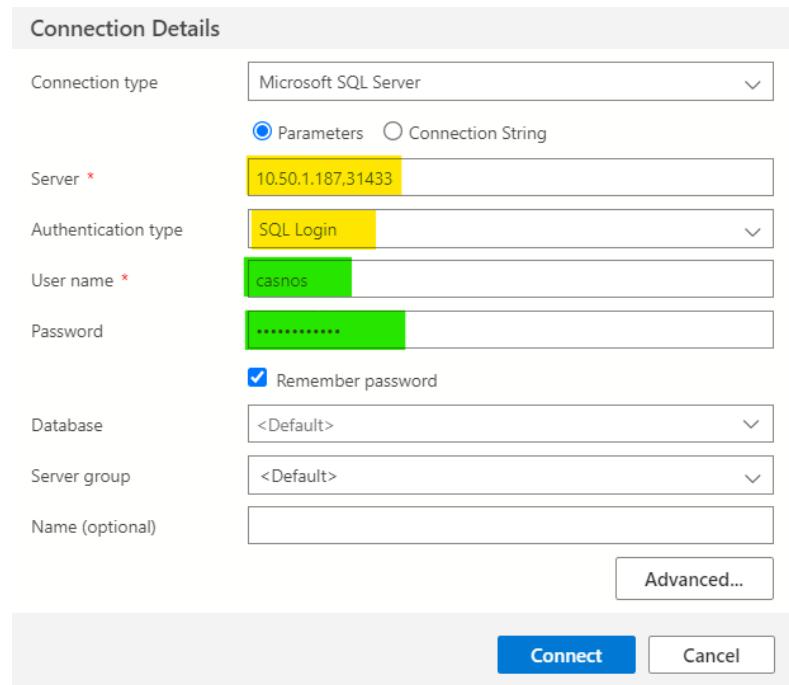


FIGURE 7.10 – Se connecter au cluster via Azure Data Studio

Ensuite, cliquer sur *Connexe* après quelques secondes nous pouvons voir que la connexions est bien établie. Voir Figure ci dessous.

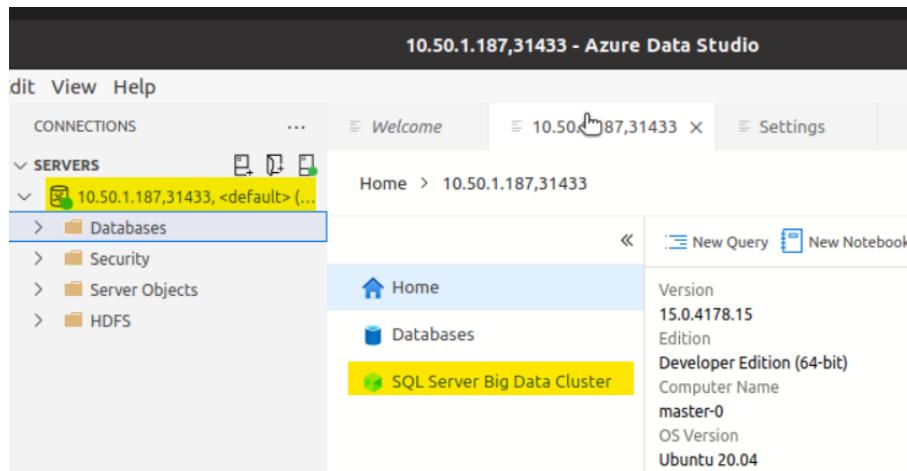


FIGURE 7.11 – Connexion au cluster via Azure Data Studio Réussie

## 7.5 Restaurer une base de données dans le cluster

L'un des avantages d'utilisation d'azure studio est la facilité d'intégration de donnée dans le cluster . Nous avons suivis les étapes ci-dessous pour restaurer la BD : "DBCASNOS"

- Appuyer sur le bouton " Restore" pour commencer le processus d'importation de la BD .

## Chapitre 7. Déploiement du big data cluster

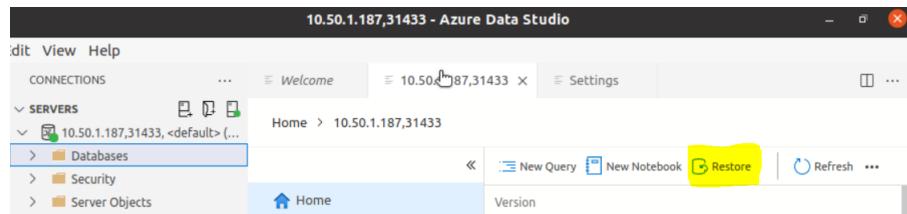


FIGURE 7.12 – Connexion au cluster via Azure Data Studio Réussie

- Copier le fichier de sauvegarde dans le conteneur SQL Server dans le pod d’instance maître du cluster Kubernetes. Après l’exécution de la commande , nous pouvons voir que le fichier .bak existe .In ne reste plus que le restaurer .

```
root@srv-k8s-master-01:/home/srv-k8s-master-01/k8s# kubectl cp /home/srv-k8s-master-01/DB_CASNOS.bak master-0:/var/tmp/DB_CASNOS.bak -c mssql-server -n mssql
```

FIGURE 7.13 – Copier le fichier de sauvegarde

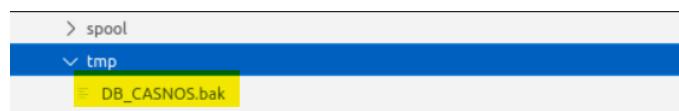


FIGURE 7.14 – Le fichier de sauvegarde DBCASNOS.BAK

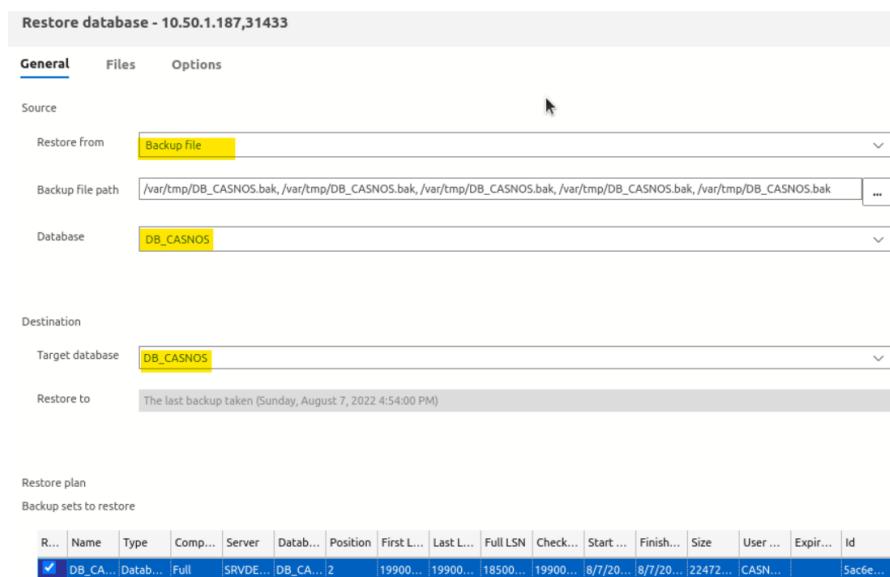


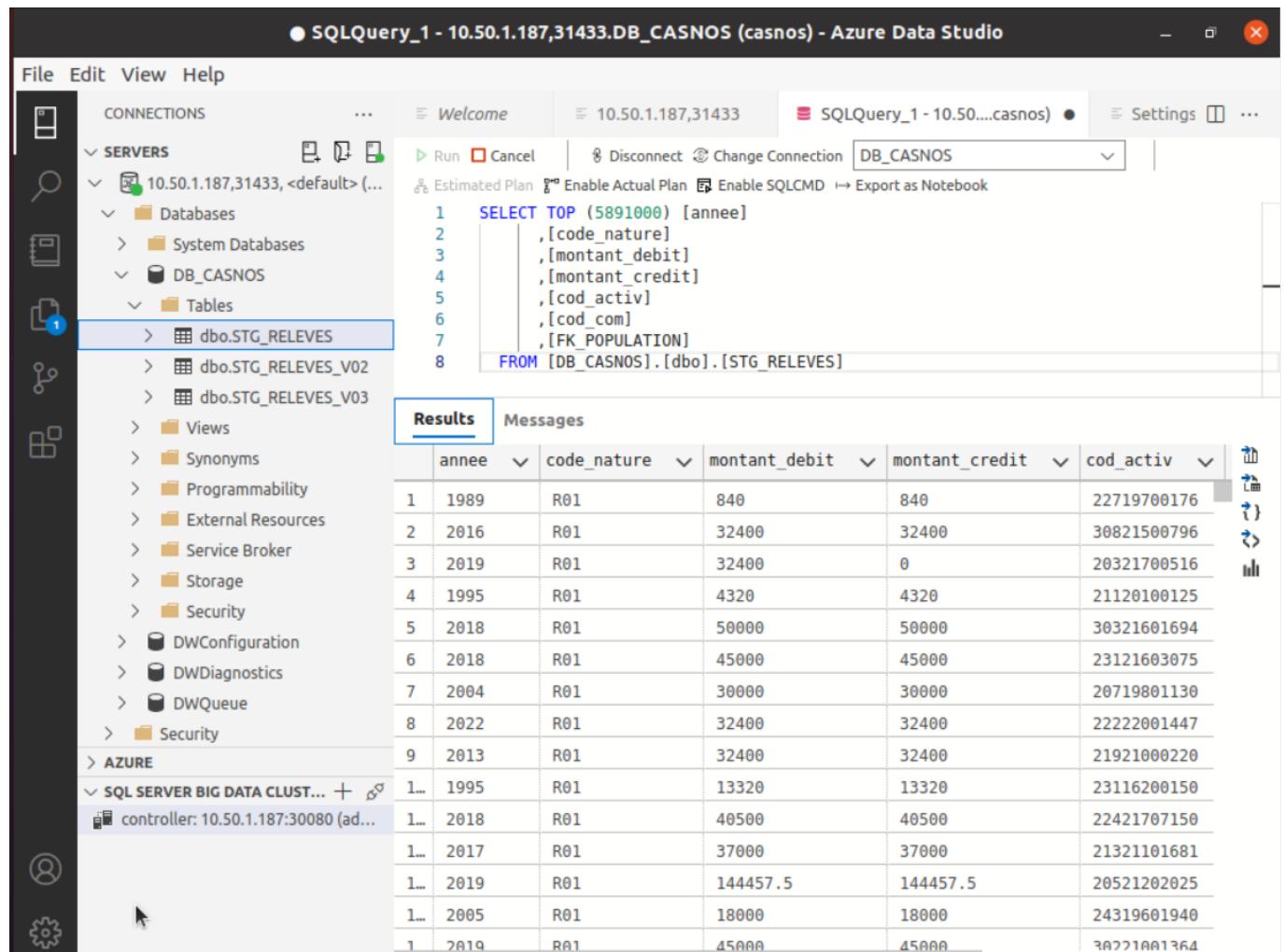
FIGURE 7.15 – Copier le fichier de sauvegarde

- Après un moment ,la Bd est bien restaurée

✓ **Restore Database succeeded** 10.50.1.187,31433 | DB\_CASNOS 5:39:37 PM - 5:43:19 PM (00:03:42)

FIGURE 7.16 – DB CASNOS bien restaurée

— Nous pouvons exécuter une requête afin de vérifier que tout est bien fait .



The screenshot shows the Azure Data Studio interface. The left sidebar displays a tree view of database connections, servers, databases, tables, and other objects. The main area shows a SQL query window with the following code:

```

SELECT TOP (5891000) [annee]
,[code_nature]
,[montant_debit]
,[montant_credit]
,[cod_activ]
,[cod_com]
,[FK_POPULATION]
FROM [DB_CASNOS].[dbo].[STG_RELEVES]

```

The results tab shows a table with the following data:

	annee	code_nature	montant_debit	montant_credit	cod_activ
1	1989	R01	840	840	22719700176
2	2016	R01	32400	32400	30821500796
3	2019	R01	32400	0	20321700516
4	1995	R01	4320	4320	21120100125
5	2018	R01	50000	50000	30321601694
6	2018	R01	45000	45000	23121603075
7	2004	R01	30000	30000	20719801130
8	2022	R01	32400	32400	22222001447
9	2013	R01	32400	32400	21921000220
1...	1995	R01	13320	13320	23116200150
1...	2018	R01	40500	40500	22421707150
1...	2017	R01	37000	37000	21321101681
1...	2019	R01	144457.5	144457.5	20521202025
1...	2005	R01	18000	18000	24319601940
1	2019	R01	45000	45000	30221001364

FIGURE 7.17 – Requête d'affichage de DB CASNOS

## 7.6 Vérifier l'état du cluster avec azure data studio

### Vue globale du cluster :

Azure data studio permet de visionner l'état de tout les composant du cluster (Voir Figure si dessous).

### Monitoring temps réel avec GRAPHANA :

Graphana est intégré avec le sql server big data cluster , il permet de visualiser en temps réel l'état du cluster et ces composants ainsi que l'état des nodes.

## Chapitre 7. Déploiement du big data cluster

The screenshot shows the 'Big Data Cluster overview' page in Azure Data Studio. The 'Cluster Properties' section indicates a 'Ready' state and 'Healthy' health status. The 'Cluster Overview' table lists seven services: SQL Server, HDFS, Spark, Control, Gateway, App, and another SQL Server instance. All services are marked as 'Ready' and 'Healthy'. The 'Service Endpoints' table provides URLs for various cluster components, such as the SQL Server Master Instance Front-End at [10.50.1.187:31433](https://10.50.1.187:31433) and the Cluster Management Service at <https://10.50.1.187:30080>.

FIGURE 7.18 – État globale du cluster avec Azure DS

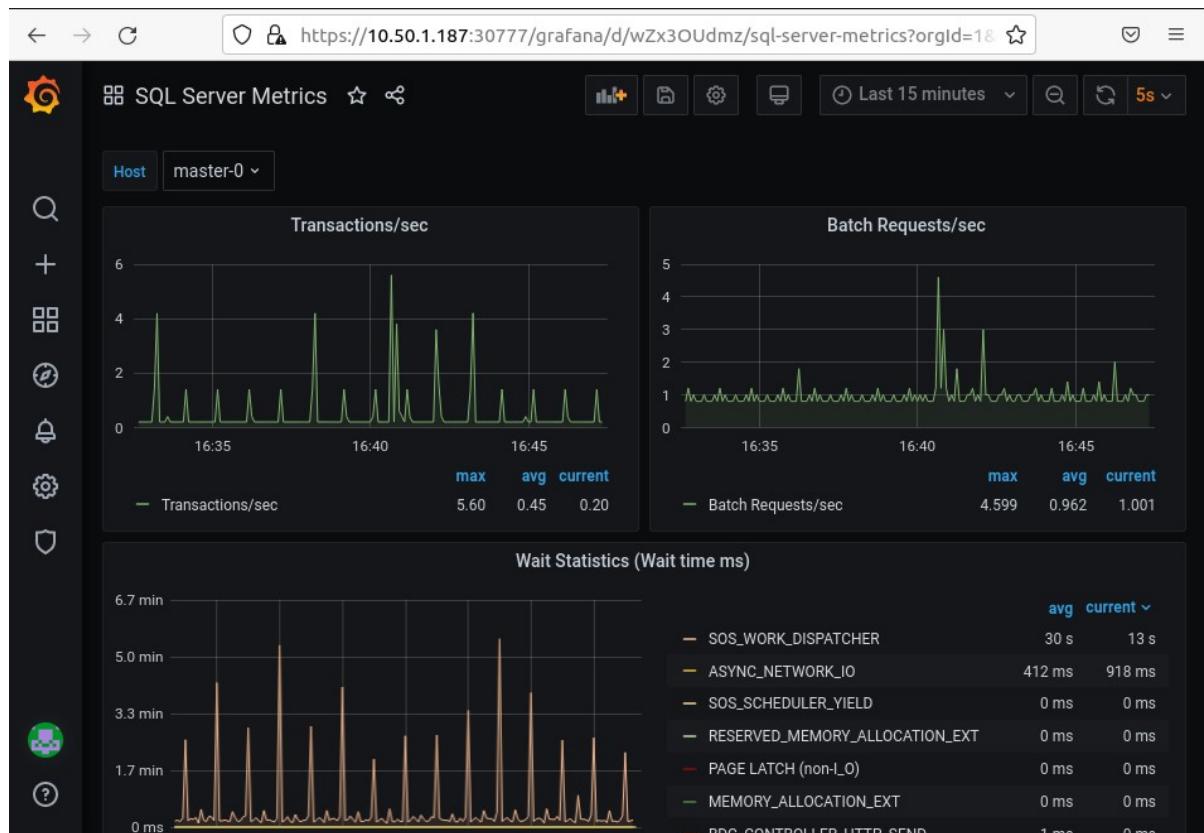


FIGURE 7.19 – État globale du cluster avec GRAPHANA

## 7.7 Vérifier l'état du cluster avec kubernetes

**Vérifier le statut du cluster :** Pour vérifier le statut du cluster, il faut d'abord se connecter au cluster avec `azdata login` puis en utilisant la commande `azdata bdc status show`. Voici le résultat de la commande .

```
root@srv-k8s-master-01:/home/srv-k8s-master-01# azdata login
Namespace: mssql-cluster
Username: casnos
Password:
Logged in successfully to `https://10.50.1.187:30080` in namespace `mssql-cluster`. Setting active context to `mssql-cluster`.
root@srv-k8s-master-01:/home/srv-k8s-master-01# azdata bdc status show
```

FIGURE 7.20 – Azdata show statut

```
Mssql-cluster: ready
=====
Services: READY
-----
Servicename      State    Healthstatus   Details
sql              READY    healthy        -
hdfs             READY    healthy        -
spark            READY    healthy        -
control          READY    healthy        -
gateway          READY    healthy        -
app               READY    healthy        -

Sql Services: ready
-----
Resourcename     State    Healthstatus   Details
master           ready    healthy        StatefulSet master is healthy
compute-0        ready    healthy        StatefulSet compute-0 is healthy
data-0            ready    healthy        StatefulSet data-0 is healthy
storage-0         ready    healthy        StatefulSet storage-0 is healthy

Hdfs Services: ready
-----
Resourcename     State    Healthstatus   Details
nmnode-0          ready    healthy        StatefulSet nmnode-0 is healthy
storage-0         ready    healthy        StatefulSet storage-0 is healthy
sparkhead         ready    healthy        StatefulSet sparkhead is healthy

Spark Services: ready
-----
Resourcename     State    Healthstatus   Details
sparkhead         ready    healthy        StatefulSet sparkhead is healthy
storage-0         ready    healthy        StatefulSet storage-0 is healthy
```

FIGURE 7.21 – Statut Globale Du Cluster

## Obtenir l'état des pods

Nous pouvons afficher l'état de tous les pods du cluster Kubernetes en exécutant la commande suivante `kubectl get pods -A` pour obtenir tous les pods et leurs états, y compris les pods qui font partie du namespace du cluster Big Data SQL Server.

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-flannel	kube-flannel-ds-6kd8h	1/1	Running	0	27h
	kube-flannel-ds-lwl74	1/1	Running	0	27h
kube-flannel	kube-flannel-ds-mdx7t	1/1	Running	0	28h
	coredns-6d4b75cb6d-mhnq	1/1	Running	0	28h
kube-system	coredns-6d4b75cb6d-qclvc	1/1	Running	0	28h
	etcd-srv-k8s-master-01	1/1	Running	2 (5h32m ago)	28h
kube-system	kube-apiserver-srv-k8s-master-01	1/1	Running	5 (4h59m ago)	28h
	kube-controller-manager-srv-k8s-master-01	1/1	Running	8 (5h ago)	28h
kube-system	kube-proxy-6fn6g	1/1	Running	0	28h
	kube-proxy-q6nfn	1/1	Running	0	27h
kube-system	kube-proxy-vr5pv	1/1	Running	0	27h
	kube-scheduler-srv-k8s-master-01	1/1	Running	8 (5h ago)	28h
kube-system	kubernetes-dashboard-cd4778d69-zkw8f	1/1	Running	0	28h
	local-volume-provisioner-j4qkl	1/1	Running	0	27h
local-storage	local-volume-provisioner-qp7pf	1/1	Running	0	27h
	appproxy-txrmm	2/2	Running	0	26h
mssql-cluster	compute-0-0	3/3	Running	0	26h
	control-xqwrz	3/3	Running	18 (4h59m ago)	27h
mssql-cluster	controlwd-4xf87	1/1	Running	14 (5h6m ago)	26h
	data-0-0	3/3	Running	0	26h
mssql-cluster	data-0-1	3/3	Running	0	26h
	gateway-0	2/2	Running	0	26h
mssql-cluster	logsdb-0	1/1	Running	0	26h
	logsui-b2pdm	1/1	Running	0	26h
mssql-cluster	master-0	3/3	Running	0	26h
	metricsdb-0	1/1	Running	0	26h
mssql-cluster	metricsdc-pfc8j	1/1	Running	0	26h
	metricsdc-sctr6	1/1	Running	0	26h
mssql-cluster	metricsui-4gdx9	1/1	Running	0	26h
	mgmtproxy-7zrch	2/2	Running	0	26h
mssql-cluster	nmnode-0-0	2/2	Running	0	26h
	sparkhead-0	4/4	Running	0	26h
mssql-cluster	storage-0-0	4/4	Running	0	26h
	storage-0-1	4/4	Running	0	26h

FIGURE 7.22 – Obtenir l'état des pods du cluster

### 7.7.1 Explication de l'organisation des namespaces

Les numéros de 1-4 corresponds aux différents namespaces du kubernetes cluster ,en ce qui suit nous allons expliquer l'utilité de chacun d'eux .

1. **Kube-flannel** : Correspond au cluster virtuel du container network interface .
2. **Kube-System** : C'est le namespace dont nous trouvons tout les composants basic de kubernetes : etcd , api-server etc .
3. **Local-storage** :Représente l'emplacement du stockage du cluster BDC .
4. **Mssql-cluster** : c'est la ou tous les pods du cluster big data se situent .

## 7.8 Conclusion

Nous avons vu dans cette partie comment créer un cluster SQL SERVER BIG DATA sur k8s.Dans la prochaine partie nous allons expliquer quelques cas d'usages .

# Cinquième partie

## Présentation des cas d'usages

# Chapitre 8

## Apprentissage automatique

### 8.1 Introduction

Ce chapitre présente le vocabulaire de l'apprentissage automatique (Machine Learning en anglais). En particulier, nous examinons comment juger de la qualité et de la performance d'un algorithme et quelles sont les précautions à prendre dans un contexte Big Data.

### 8.2 L'apprentissage Automatique

« Le Machine Learning (ML) est une forme d'intelligence artificielle (IA) qui est axée sur la création de systèmes qui apprennent, ou améliorent leurs performances, en fonction des données qu'ils traitent.»

Le Machine Learning (ou apprentissage automatique) est une branche de l'IA qui utilise des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données et d'améliorer leurs performances dans l'exécution d'une tâche spécifique.

Il concerne plus précisément, à l'analyse, l'optimisation, le développement et l'implémentations de telles méthodes. [6]

### 8.3 Les types d'apprentissage automatique

On distingue différents types d'algorithmes Machine Learning. Généralement, ils peuvent être répartis en trois catégories : apprentissage supervisé, non supervisé et par Renforcement .

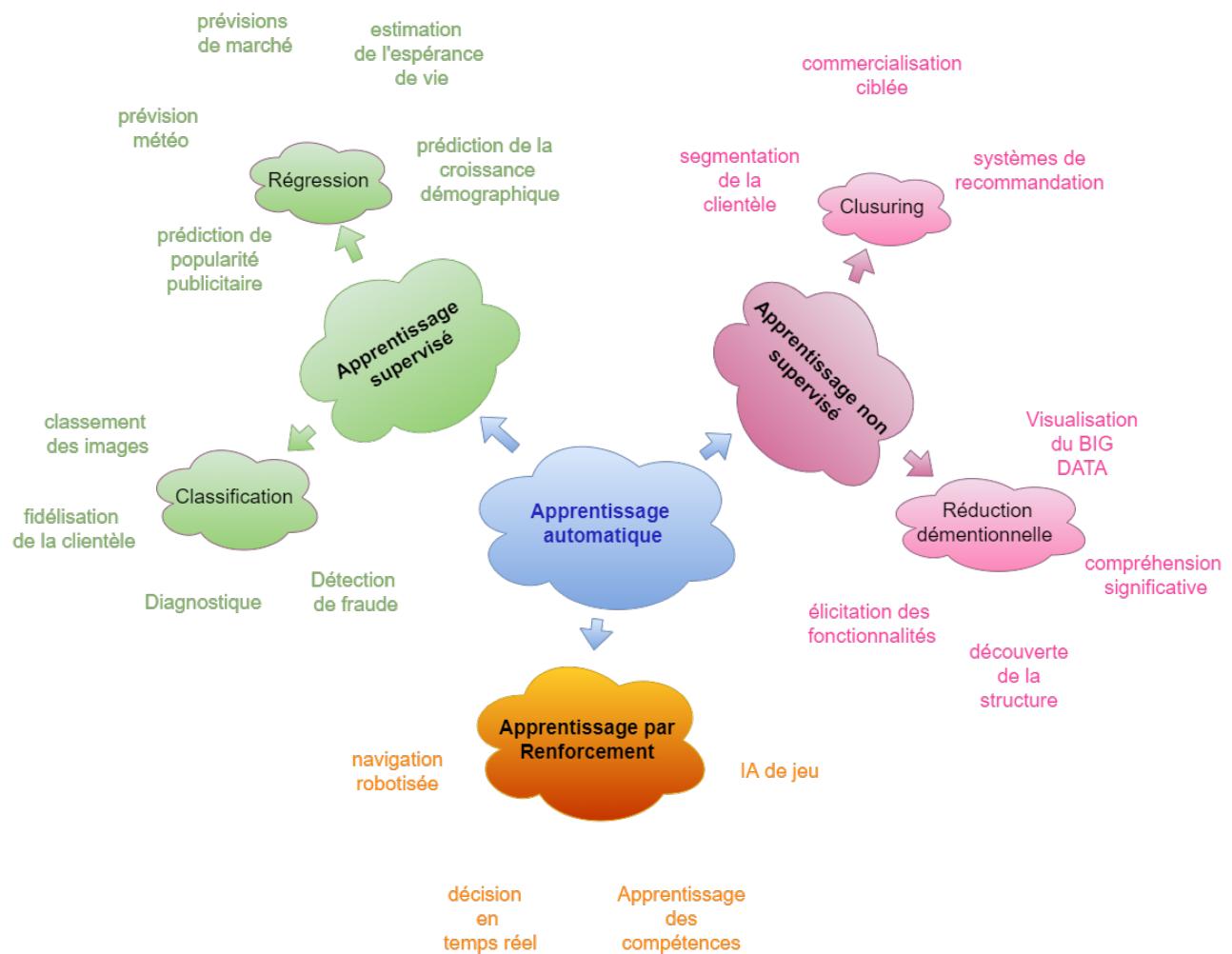


FIGURE 8.1 – Les différentes types d'apprentissage automatique

### 8.3.1 Apprentissage supervisé

L'apprentissage automatique supervisé s'entraîne sur une base de données étiquetées. En d'autres termes, les données sont étiquetées avec des informations que le modèle est destiné à déterminer pour l'apprentissage automatique.

L'apprentissage automatique supervisé facilite l'apprentissage en nécessitant moins de données d'apprentissage que les autres méthodes d'apprentissage automatique, car les résultats du modèle peuvent être comparés aux résultats réels étiquetés.

L'apprentissage supervisé tente de répondre à deux questions :

- \* Classification : « quelle classe ? » ;
- \* Régression : « combien ? ».

#### 8.3.1.1 Classification

Un problème de classification se pose lorsque la variable de sortie est catégorielle, comme « rouge » ou « bleu » ou « maladie » et « aucune maladie ».

La classification est définie comme le processus de reconnaissance, de compréhension et de regroupement d'objets et d'idées dans des catégories prédéfinies appelées «classes».

Elle prédit les étiquettes de classe catégorielles en fonction de l'ensemble d'apprentissage et des valeurs de la variable cible et les utilise dans de nouvelles données de classification.

Il existe plusieurs modèles de classification comme par exemple la régression logistique, l'arbre de décision, la forêt aléatoire, l'arbre de gradient et Naïve Bayes.[7]

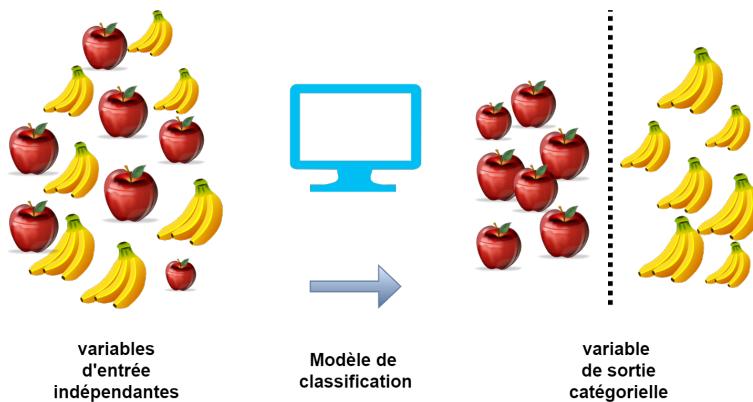


FIGURE 8.2 – Exemple de classification

### 8.3.1.2 Régression

Un problème de régression survient lorsque la variable de sortie est une valeur réelle ou continue, telle que "salaire" ou "poids".

Le but de la régression est l'estimation de valeur de sortie (numérique) à partir des valeurs d'un ensemble de caractéristiques d'entrée. Par exemple, l'estimation de prix d'une maison est sur sa surface, son nombre d'étages, sa localisation, etc. Le problème revient alors à estimer une fonction de calcul basée sur des données d'entraînement.[1]

### 8.3.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé est utilisé pour découvrir des structures et des modèles dans les données sans référent aux résultats étiquetés.

Deux principales méthodes sont utilisées dans l'apprentissage non supervisé, comprenant le clustering et la réduction de dimensionnalité.

#### 8.3.2.1 Clustering

Le clustering est une méthode d'apprentissage automatique non supervisée permet de séparer les données en fonction de leurs propriétés ou fonctionnalités et de les regrouper en différents clusters en fonction de leurs similitudes.

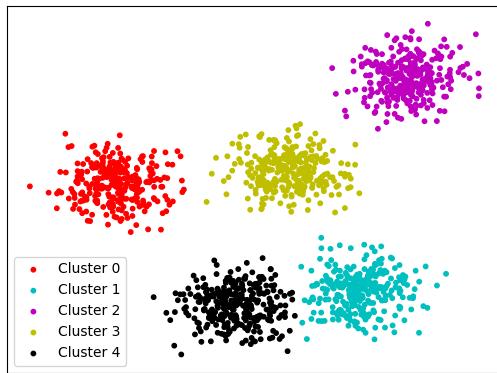


FIGURE 8.3 – Exemple de prédiction de 5 clusters

Le but des algorithmes de clustering est de donner un sens aux données et d'extraire de la valeur à partir de grandes quantités de données non étiquetées .[3]

### 8.3.2.2 Réduction dimensionnelle

La réduction de dimensionnalité est un processus qui consiste à réduire le nombre de variables dans un ensemble de données ou bien de les projeter d'un espace de grande dimension vers un espace de dimension plus petit.

En effet, la réduction du nombre de variables dans les données d'apprentissage augmente la robustesse ou la stabilité de l'algorithme. L'apprentissage automatique peut être amélioré en créant des modèles plus simples en supprimant les variables inutiles.

Certains des algorithmes de réduction de dimensionnalité les plus populaires sont :

- **ACP** (Analyse en Composantes Principales) : L'Analyse en Composantes Principales consiste à identifier les directions principales et les variantes importantes.
- **LDA** (linear discriminant analysis) : l'analyse discriminante linéaire identifie les directions décorrélées les unes des autres.

### 8.3.3 Apprentissage par renforcement

Ce type d'apprentissage commence par un agent (algorithme) effectuant une sélection dans une liste d'actions. Selon l'action choisie, il recevra des commentaires de l'environnement (soit d'un humain, soit d'un autre algorithme dans certaines circonstances) : C'est soit une récompense pour les bons choix, soit une punition pour les mauvaises actions.

L'agent (algorithme) apprend la stratégie (ou le choix d'action) qui maximise la récompense cumulée. Ce type d'apprentissage est couramment utilisé dans le contexte de la robotique, de la théorie des jeux et des véhicules autonomes.[5]

## 8.4 Principaux algorithmes du Machine Learning

### 8.4.1 Arbres de décision

Un arbre de décision est un modèle de prédiction ou de classification utilisé en apprentissage supervisé. Un arbre de décision est un classificateur obtenu en partitionnant de manière récursive un ensemble de données et en prédisant un seul attribut dans chaque partition.

L'arbre de décision permet d'évaluer différentes actions possibles en fonction des bénéfices, des probabilités et des coûts en basant sur un ensemble de données exploitable. Elle est également utilisée pour créer des algorithmes d'apprentissage automatique pour déterminer mathématiquement le meilleur choix dans une situation donnée. Ce modèle bien connu a conduit à l'apparition des algorithmes puissants tels que XGBoost et Random Forest.

En théorie des graphes, un arbre est un graphe non orienté, acyclique (ne contient aucun cycle) et connexe (un graphe est connexe quand tout sommet peut être relié à tout autre sommet soit par une arête ou par une suite d'arêtes). Dans l'arbre de décision, nous trouvons :

- Nœud racine : (l'accès à l'arbre se fait par ce nœud) la première séparation des données d'apprentissage se fait dans ce nœud.
- Nœud interne : concerne un attribut qui permet de répartir les éléments de manière homogène entre les différents fils de ce nœud .
- Nœuds terminaux (feuilles) : nœuds qui n'ont pas de descendants, représentant les prédictions sur les données à classer.
- Les branches : faire le lien entre un nœud et ses enfants, qui représentent les valeurs de l'attribut de nœud.

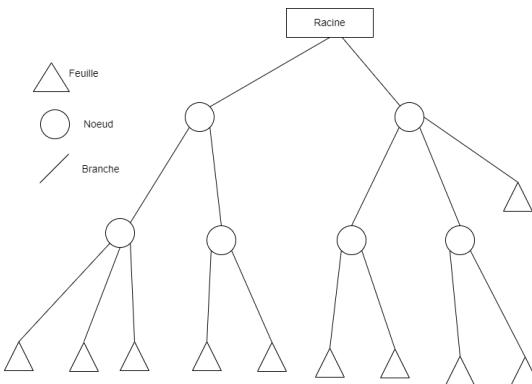


FIGURE 8.4 – Illustration d'une arbre de décision

L'arbre de décision est une catégorie d'arbres utilisée dans l'exploration de données et en informatique décisionnelle. Ils emploient une représentation hiérarchique de la structure des données sous forme des séquences de décisions (tests) en vue de la prédiction

d'un résultat ou d'une classe.

Chaque observation attribuée à une classe, est décrite par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prise dans les nœuds feuille.

Chaque feuille représente soit une valeur de la variable cible, soit une distribution de probabilité des diverses valeurs possibles de la variable cible.

La méthode est non paramétrique et ne fait aucune hypothèse a priori sur la distribution des données.

### **8.4.2 Random forest**

L'objectif de l'algorithme de forêt aléatoire est de conserver la plupart des points forts des arbres de décision tout en éliminant les inconvénients des arbres de décision, en particulier la sensibilité au sur-apprentissage.

Random Forest est un algorithme qui construit des centaines d'arbres de décision à partir du même jeu de données. Ces modèles sont ensuite combinés pour obtenir un modèle unique.

Le modèle résultant n'est pas un arbre unique, mais un groupe d'arbres formant une forêt. Chaque arbre est construit à partir d'un échantillon aléatoire de l'ensemble de données.

Son principe est très simple, dans une forêt aléatoire, chaque arbre de décision donne une prédiction, et le résultat final est obtenue selon le type du problème :

- Classification : faisant un vote pour choisir la classe la plus présente.
- Régression : calculant le moyen des prédictions.

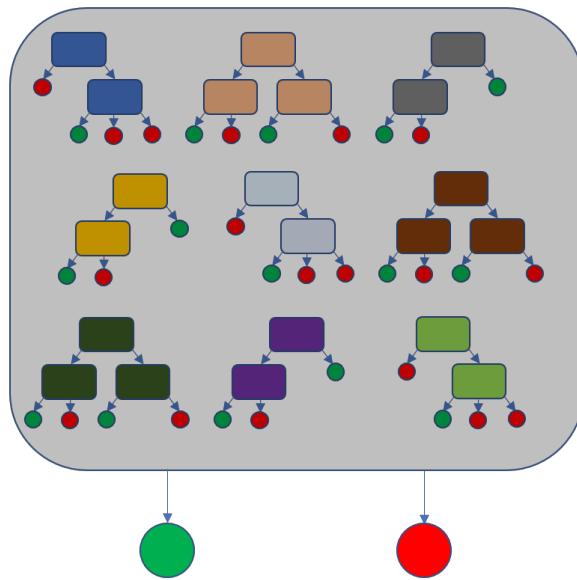


FIGURE 8.5 – Le processus d'une Forêt aléatoire

### **Bagging : méthode ensembliste parallèle**

Le Bagging est une technique en intelligence artificielle qui consiste à assembler un grand nombre d'algorithmes avec de faibles performances individuelles pour en créer un algorithme beaucoup plus efficace. Les algorithmes de faible performance sont appelés les « weak learners » et le résultat obtenu « strong learner ». L'idée derrière cet algorithme est que plusieurs petits algorithmes fonctionnent mieux qu'un seul grand algorithme.

Les principes de bagging incluent :

1. Créez de nombreux sous-échantillons aléatoires de notre ensemble de données. Cela pour chaque modèle on aura un ensemble de donnée.
2. Entraîner chaque modèle sur la partie aléatoire pour récréer les prédictions.
3. Le résultat avec le plus de vote devient le résultat finale de notre modèle. (cas regression, on prend la moyenne).

Les modèles dans ce cas sont entraîné simultanément.

### **Boosting : méthode ensembliste séquentielle**

Le boosting est une méthode utilisée dans le machine learning pour réduire les erreurs dans l'analyse prédictive de données. Le boosing fonctionne comme suit :

1. La première étape consiste à créer le premier modèle de base à partir de l'algorithme sélectionné. Les observations sont attribuées avec des poids égaux . D'après les résultats obtenus avec ce modèle, si une observation est mal classée, cela augmente son poids.
2. Ensuite , à l'aide des données pondérées obtenues lors de la première étape on construire le deuxième modèle afin de corriger les erreurs de modèle précédent.
3. Ce processus se répète jusqu'à ce que l'ensemble de données d'apprentissage soit correctement prédit ou que le nombre maximal de modèles soit ajouté.

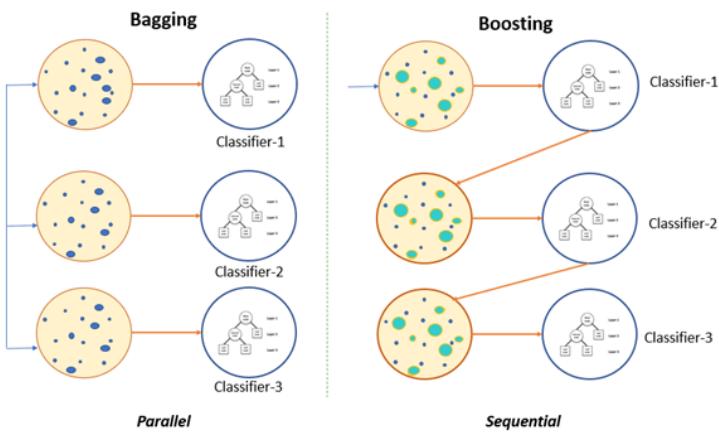


FIGURE 8.6 – Bagging vs Boosting

### 8.4.3 XGBoost(eXtreme Gradient Boosting)

C'est une implémentation d'arbres de décision boostés par gradient conçus pour la vitesse et la performance qui se base sur les méthodes ensemblistes bagging et boosting. XGBoost est un modèle de machine Learning basé sur l'apprentissage d'ensemble séquentiel et les arbres de décision.

Il utilise le boosting de gradient. XGBoost traite les données en plusieurs morceaux compressés, ce qui rend le tri et le traitement parallèle beaucoup plus rapides.

## 8.5 Indicateurs de performances

Les indicateurs de qualité du modèle sont basés sur des éléments de matrice de confusion. La matrice de confusion est un outil de mesure de la performance des modèles de classification à 2 classes ou plus. Dans le cas binaire (i.e. à deux classes, le cas le plus simple), la matrice de confusion est un tableau à 4 valeurs représentant les différentes combinaisons de valeurs réelles et valeurs prédites comme dans la figure ci-dessous.

		Les valeurs réelles	
		Positive	Négative
Les valeurs prédictées	Positive	TN Ceci n'est pas une pomme	FN Ceci n'est pas une pomme
	Négative	FP C'est une pomme	TP C'est une pomme

FIGURE 8.7 – Matrice de confusion

Les 4 cas possibles lors d'une prédiction binaire sont (prenant l'exemple de fruits) :

- **Vrai négatif** (True negative) :

- La prédiction est négative et c'est la réalité.
- On prédit que ce fruit "une banane" n'est pas une pomme , ce qui est vrai.

- **Vrai positif** (True Positive) :

- La prédiction est positive et c'est la réalité.
- On prédit que ce fruit "une pomme" est une pomme, c'est le cas.

- **Faux positif** (False positive) :

- La prédiction est positive mais ce n'est pas la réalité.
- On prédit que ce fruit "une banane" c'est une pomme , c'est faux.

- **Faux négatif** (False negative) :

- La prédiction est négative mais ce n'est pas la réalité.
- On prédit que ce fruit "une pomme" n'est pas une pomme , c'est faux.

Cette matrice est indispensable pour définir les différentes métriques de classification telles que l'Accuracy, le F1-score ou encore l'AUC, PR et l'AUC ROC.[4]

### 8.5.1 Précision

c'est le taux de positifs bien prédits parmi tous les positifs prédits.

$$precision = \frac{VP}{VP + FP} \quad (8.1)$$

### 8.5.2 La spécificité

La probabilité qu'un individu négatif est classé négatif par le modèle

$$spcifit = \frac{VN}{VN + FN} \quad (8.2)$$

### 8.5.3 La sensibilité(Recall)

La probabilité qu'un individu positif est classé positif par le modèle.

$$Recall = \frac{VP}{VP + FN} \quad (8.3)$$

### 8.5.4 L'exactitude (Accuracy)

La précision est la mesure traditionnelle la plus largement utilisée pour l'évaluation modèle, il représente le pourcentage d'exemples bien classés.

$$Accuaracy = \frac{VP + VN}{VP + VN + FN + FP} \quad (8.4)$$

### 8.5.5 Le taux d'erreurs(error rate)

Le taux d'erreur est un pourcentage d'exemples mal classé.

$$ER = 1 - Accuaracy \quad (8.5)$$

### 8.5.6 F1-score

La moyenne pondérée de la précision et la sensibilité.

$$F1 - score = 2 * \frac{Recall * precision}{Recall + precision} \quad (8.6)$$

### 8.5.7 ROC-AUC

La courbe ROC est une représentation graphique de la relation existante entre la sensibilité et la spécificité d'un test pour toutes les valeurs seuils possibles.

La zone sous la courbe ROC (Area under curve, AUC) est un indicateur récapitulatif du rendement de la courbe ROC qui peut résumer le rendement d'un classificateur en une seule mesure. AUC est une valeur discrète qui peut être comparée entre les classificateurs.

Une AUC de 0.5 indique que le classificateur ne performe pas mieux qu'un classificateur aléatoire tandis qu'une AUC de 1 indique un classificateur parfait. Pour pouvoir prendre en considération une valeur AUC, il faut donc qu'elle soit supérieure à 0.5.

## 8.6 Conclusion

Dans ce chapitre, nous avons traité l'apprentissage automatique, ses types et ses méthodes les plus courantes, ainsi que les algorithmes les plus courants. Nous avons aussi décrit les indicateurs de performances qu'on va l'utiliser par la suite pour évaluer notre modèle.

# Chapitre 9

## Conception

### 9.1 Introduction

Chaque année l'organisme CASNOS fait des prévisions en matière de dépense et en matière de réalisation d'une recette. Les prévisions n'ont jamais été basé sur des critères fixes. Par exemple, l'année précédente, la prévision était basée sur le potentiel : Registre commercial vu qu'il représente 90% de la population, donc généralement il pourra être comme un premier potentiel sur lequel les prévisions seront faites. La recette d'une année donnée représente 15% de montant débit total des adhérents, chaque adhérent déclare son montant débit lors de son affiliation au CASNOS.

A l'effet de certaines raisons comme les clients qui s'arrêtent de se présenter aux agences de CASNOS pour payer leurs cotisations, ou bien l'irrégularité de paiement de clients, la recette calculée ne concerne pas que les adhérents qui se présentent pour payer leur cotisation, mais même ceux qui ne paient pas, ce qui rend la prévision incorrecte. Le

machine learning aide à mieux exploiter les données des clients et ainsi effectuer une segmentation plus fine et en temps réel. Contrairement aux outils traditionnels d'analyse de données, le machine learning s'avère plus efficace en termes de précision et de vitesse. Plus il dispose de données, plus il peut apprendre et proposer des solutions à des insights de plus en plus pertinents. Pour cela, nous proposons d'élaborer un modèle de machine learning pour segmenter les adhérents de CASNOS en deux classes, une classe des clients fidèles et une classe des clients infidèles, afin de savoir le nombre de cotisants qui vont se rapprocher pour payer leur cotisation dans une année donnée et prédire la recette annuelle.

Dans ce chapitre, nous allons présenter les étapes de conception de notre solution. Nous décrivons le processus de notre travail en suivant le processus CRISP-DM, on commence par la collection des données, passant par la compréhension et la préparation des données puis la construction du modèle prédictive et par la suite, l'évaluation et le déploiement de notre modèle.

### 9.2 Solution proposée

Notre but est d'élaborer un modèle pour prédire les clients fidèles (c.-à-d., ceux qui vont se rapprocher aux agences pour payer leur cotisation dans une telle année) à partir

des observations passées.

Afin d'atteindre ce but, nous avons élaboré un modèle de prédiction de nombre de cotisants en suivants les étapes suivantes :

- Détermination du type de problème.
- Obtenir le jeu de données.
- Choix du modèle.
- Faire les testes et le déploiement.

Notre cas d'étude est un cas de classification binaire, en effet ? il existe deux types d'adhérents : Adhérent infidèle(0), Adhérent fidèle (1).

### 9.3 Collecter les données

Nous avons reçu un Fichier BAK (.bak) qui contient notre base de données sur laquelle notre travail est fait.

Le jeu de données qu'on a utilisé est un ensemble de données étiqueté correspond à la cotisation des adhérents durant l'années actuelles « 2022 ». Ce jeu de données contient 2 201 188 lignes et 10 colonnes.

La figure suivante décrit les données qui serviront pour la prédiction de cotisants :

Variable	Type	Codification	Description
<b>POPULATION_KEY</b>	Continue	Numérique	L'ID de l'adhérent
<b>GROUPE_CSP_FR</b>	Qualitative	PROFESSION LIBERALE, COMMERCANT, ARTISAN, SOCIETE, AGRICOLE	Catégorie sociaux professionnelle, indique la nature de l'activité principale que l'adhérent exerce
<b>POSITION_KEY</b>	Continue	1,2,3	La position de l'adhérent . 1:affilié actif, 2: retraité actif , 3:affilié en veille
<b>cod_com</b>	Continue	[0101,...,5999]	C'est le code de commune ou l'adhérent exerce sa activité principale
<b>dat_debut</b>	Continue	datetime	La date début de l'activité principale actuelle
<b>taux_fidelite</b>	Continue	Numérique	Le taux de fidélité d'un adhérent correspond au rapport entre le nombre d'année cotisé et le nombre d'année de l'activité principale actuelle
<b>nbr_annee_activite</b>	Continue	Numérique	C'est le nombre d'années depuis la date début d'activité jusqu'à la date actuelle
<b>nbr_annee_cotise</b>	Continue	Numérique	c'est le nombre d'années dont l'adhérent a payé sa cotisation
<b>nbr_activite</b>	Continue	Numérique	C'est le nombre d'activité qu'il exerce l'adhérent
<b>annee_en_cours</b>	Qualitative	0,1	si l'adhérent a payé sa cotisation dans l'année actuelle ou pas.

FIGURE 9.1 – Description des colonnes de Dataset

- Chaque lignes de jeu de données correspond à un adhérent (2201188 Adhérents).
- Un adhérent peut exercer plusieurs activités, dans le cas où il n'y a pas une seule activité, l'adhérent choisit une pour qu'elle soit son activité principale.
- Il existe cinq catégories d'activités.
- Notre étude s'intéresse qu'au trois type d'adhérent (affilié active, retraité active, affilié en veille).

## 9.4 Exploration des données

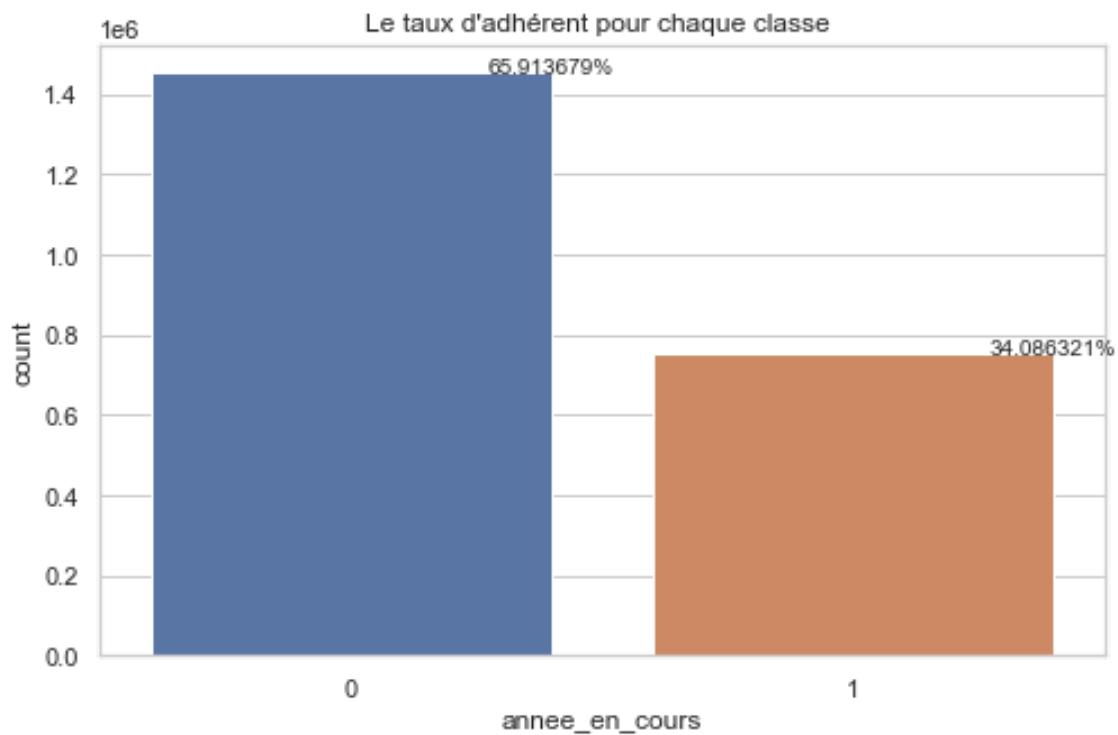


FIGURE 9.2 – Le taux d'adhérent pour chaque classe

D'après la figure précédente on constate que notre jeu de données n'est balancé vu que le taux des adhérents qui n'ont pas cotisé est presque le double de taux des adhérents qui ont cotisé.

## Chapitre 9. Conception

---

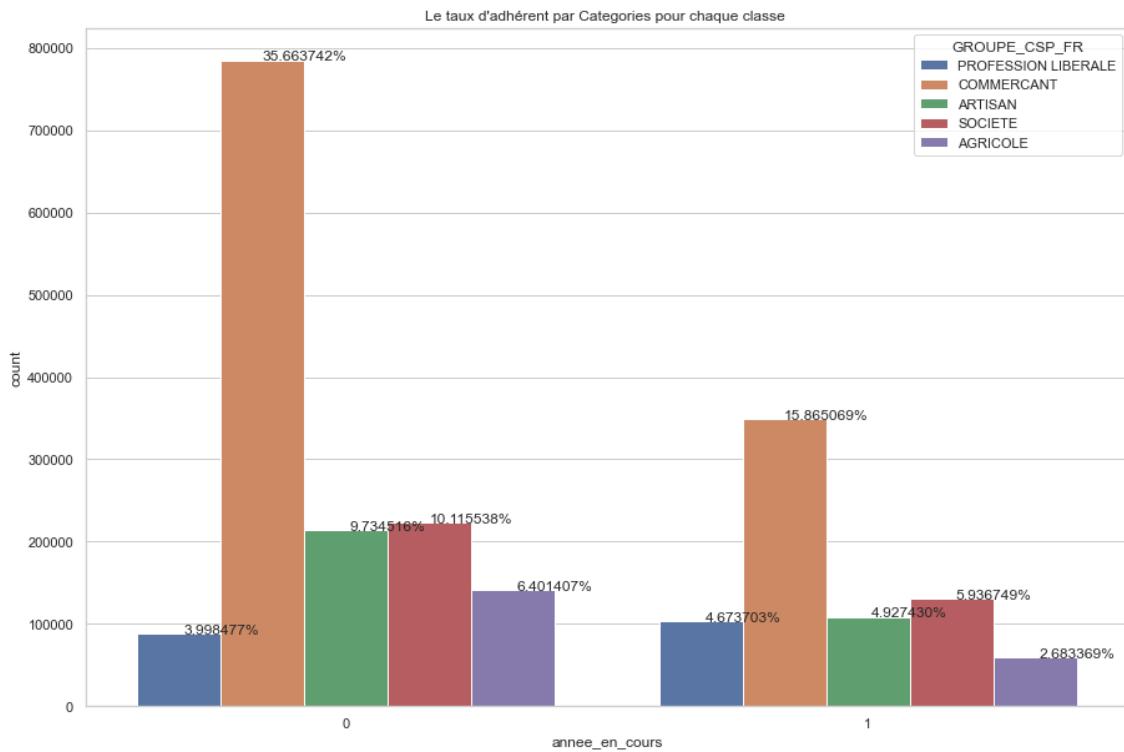


FIGURE 9.3 – Le taux d'adhérent pour chaque classe par catégorie

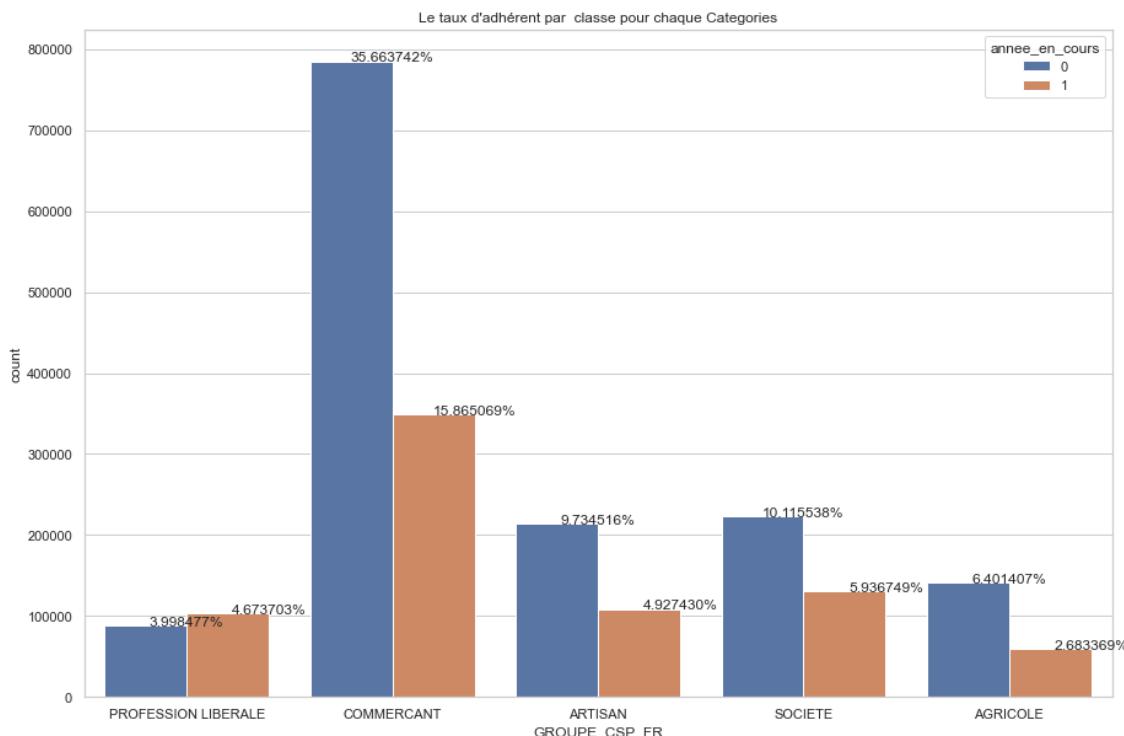


FIGURE 9.4 – Le taux d'adhérent pour chaque catégorie par classe

D'après les figures précédentes, on peut conclure les points suivants :

- La catégorie la plus marquante des adhérents est « commerçant ».

- On remarque que la majorité des clients exerçant des métiers artisanaux paient leur cotisation.
- Alors que pour les autres catégories, la plupart des clients ne paient pas leur cotisation.
- On note que pour les retraités actifs, leur taux de cotisation est de 1.62%, tandis que le pourcentage de ceux qui n'ont pas payé est de 0.88%.
- Pour les affiliés actives, la plupart ne paient pas leurs cotisations avec une différence de 15%.

## 9.5 Préparation des données

Le pré-traitement de données regroupe les différents changements appliqués sur l'ensemble de données. L'un des principaux avantages de la mise en place d'un processus formel de préparation des données est que les utilisateurs peuvent consacrer moins de temps à la recherche de données et les structurer. Le pré-traitement appliqué sur notre ensemble de données est :

- Changement de type de variable : par exemple les dates de type "object" vers le type "date".
- Traitement des valeurs manquantes : en supprimant ces valeurs nuls ou bien les remplacer avec une autre valeur significative.

POPULATION_KEY	0
POSITION_KEY	0
GROUPE_CSP_FR	0
cod_com	94
dat_debut	169553
nbr_annee_activite	169553
nbr_annee_cotise	0
nbr_activite	0
annee_en_cours	0
taux_fidelite	169553
<b>dtype:</b>	<b>int64</b>

FIGURE 9.5 – Les valeurs nulles dans le jeu de données

- L'encodage numérique de variable "GROUPE-CSP-FR" de type catégorique.

```
data['GROUPE_CSP_FR'].unique()
0.2s
array(['PROFESSION LIBERALE', 'COMMERCANT', 'ARTISAN', 'SOCIETE',
       'AGRICOLE'], dtype=object)
```

FIGURE 9.6 – Les catégories des activités

- Suppression des valeurs non significatives : comme les dates supérieur à la date actuelle.

```
data['dat_debut'].max()
✓ 0.6s
Timestamp('2088-10-21 00:00:00')
```

FIGURE 9.7 – La date maximale de début de l'activité d'un adhérent

## 9.6 Traitement des classes déséquilibrées

Après l'analyse de données , on voit que notre data set est composé de deux classes :

- Classe des clients fidèles ( $y=1$ ) avec un taux de 34.09%.
- Classe des clients infidèles ( $y=0$ ) avec un taux de 65.91%.

Comme on a mentionné auparavant, notre ensemble de données est déséquilibrés, on aura de mauvaises performances sur les classes minoritaires. Dans notre cas, la classe minoritaire « 1 » est la plus importante pour différencier les clients fidèles de clients infidèles. Pour traiter ce problème, on a essayé l'un des algorithmes de sur-échantillonnage (oversampling) : Synthetic Minority Oversampling Technique (SMOTE) sur les données d'apprentissage afin d'augmenter la taille des classes minoritaires. SMOTE est un algorithme

de sur-échantillonnage crée des observations synthétiques basées sur les observations des classes minoritaires existantes. Dépend de la quantité de sur-échantillonnage nécessaire, SMOTE calcule les k plus proches voisins pour créer des exemples synthétiques.

## 9.7 Modèle de prédiction

Nous avons appliqué trois algorithmes sur notre data set pour prédire les cotisants fidèles, l'architecture de chacune de ces approches a été modifiée et améliorée en essayant plusieurs paramètres pour chaque algorithme.

Ces algorithmes sont : Random forest, décision tree et XGBOOST.

- Les paramètres de l'algorithme de Random Forest :  
`random_state=42`, cet paramètre est un nombre aléatoire permet juste de réutiliser les mêmes tirages aléatoires.  
`n_jobs=-1`, est un entier, spécifiant le nombre maximum de noeuds de calcul exécutés simultanément, -1 indique tous les processeurs sont utilisés  
`max_depth=20`, représente la profondeur de chaque arbre de la forêt.  
`n_estimators=600`, ce paramètre spécifie le nombre d'arbre dans le modèle.
- Les paramètres de l'algorithme de decision tree : par défaut
- Les paramètres de l'algorithme de XGBoost :  
`n_estimators=100`.  
`max_depth=20`.  
`min_child_weight=5`, est donnée comme suit : c'est la somme minimale du poids d'instance nécessaire dans un enfant.

## **9.8 Le modèle optimale**

Après l'évaluation de ces modèles et la comparaison des résultats, on choisit le meilleur en terme des indicateurs de performances et le temps d'exécution. Enfin après avoir choisi le modèle optimal, nous enregistrons cette solution dans un fichier et la présentons à l'entreprise.

Après avoir évaluer le modèle, on le déploie dans notre plateforme pour l'utiliser.

## **9.9 Conclusion**

Dans ce chapitre, nous avons présenté la conception de notre solution, en détaillons les différentes étapes. Nous présentons la mise en œuvre de cette solution dans le chapitre suivant.

# Chapitre 10

## Implémentation

### 10.1 Introduction

Ce chapitre sera consacré à la construction de notre modèle , l'implémentation est réalisée en utilisant le langage Python

### 10.2 Mise en place de l'environnement

#### 10.2.1 Visual Studio Code

Visual Studio Code est un éditeur de code simplifié, qui est gratuit et développé en open source par Microsoft. Il fonctionne sous Windows, mac OS et Linux.



#### 10.2.2 Anaconda

Anaconda est une distribution gratuite et open source des langages de programmation Python et R appliqués au développement d'applications de science des données et d'apprentissage automatique, simplifiant la gestion et le déploiement des packages.



#### 10.2.3 Jupyter Notebook

Notre modèle sera développé sur Jupyter Notebook. C'est une application web open source permet de regrouper au sein d'un même document du code exécutable en direct mais aussi du texte narratif, des équations et des visualisations.



#### 10.2.4 Python

Python est le langage le plus populaire dans le monde de l'intelligence artificielle. Python est très utilisé au sein de la communauté scientifique et particulièrement dans le domaine de l'intelligence artificielle.



### 10.2.5 Numpy

NumPy est une bibliothèque pour le langage de programmation Python destinée à manipuler des matrices et des tableaux multidimensionnels, et des fonctions mathématiques manipulant ces tableaux.



### 10.2.6 scikit-learn

Scikit-learn est une bibliothèque d'apprentissage automatique Python open source. Elle fournit un ensemble d'outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression et le clustering.



### 10.2.7 Pandas

Pandas est une bibliothèque open source écrite pour le langage de programmation Python qui permet la manipulation et l'analyse de données. En particulier, il fournit des structures de données et des opérations pour manipuler des tableaux numériques et des séries temporelles.



## 10.3 Compréhension des données

### 10.3.1 Importation des données

A l'aide de **Pandas**, nous avons importer notre jeu de données à l'aide de bibliothèque **pyodbc** qui permet de se connecter au serveur de SQL Server.

Avant de se connecter à un serveur Microsoft SQL, nous avons d'abord besoin de quelques détails sur le serveur : le nom du pilote, le nom du serveur et le nom de la base de données.

Avec ces informations, une chaîne spéciale doit être créée, qui sera passée à la fonction `connect()` de la bibliothèque pyodbc. Puis nous allons utiliser la fonction `read_sql()` pour récupérer la table SQL associé au notre jeu de données et le transformer en un tableau de type pandas DataFrame.

## 10.4 Nettoyage des données

Dans cette étape nous avons appliqué un ensemble de tâches afin de préparer notre jeu de données.

- Suppressions des valeurs manquantes
- Transformation de type de objets vers le type date pour la colonne "Date début".
- Suppression des dates dont l'année est supérieure à 2022. (Ce genre d'erreurs est une erreur de saisie)
- Suppression des codes communes inférieur à 0101, (les 2 premiers chiffres correspond au numéro de wilaya, par exemple 3502, veut dire la commune de l'activité principale de cet adhérent se situe dans la wilaya de Boumerdès

- Suppression des codes communes supérieur à 5999. (il existe 59 wilaya en algérie, donc on suppose que la commune maximale est la commune avec l'ID 5999 et toute commune dont son ID est supérieur à 5999 on la supprime.
- L'encodage de colonne groupe de catégories sociaux.

## 10.5 Partition des données

On commence par la division de notre jeu de données en deux ensembles :

- un ensemble X contient l'ensemble des données des variables prédictives,
- Y contient la variable cible.

Ensuite on définit l'ensemble d'entraînement et l'ensemble de test en utilisant la fonction `train_test_split` de la librairie `sklearn` en attribuant 80% à l'ensemble d'entraînement et 20% pour l'ensemble de test.

## 10.6 Évaluation

L'évaluation de modèles se base sur les métrics , recall précision, f1-score pour chaque classe et l'accuracy :

- Si nous voulons étudier quel type d'utilisateurs sont plus susceptibles à cotiser, alors nous aurons besoin d'un bon f1-score.
- Si nous voulons trouver tous les clients à risque de ne pas cotiser et essayer de les faire rester, alors le taux de rappel devient plus important, d'où un faible rappel signifie que nous ne pouvons pas capturer tous les clients à risque.

Les figures suivantes représentent la comparaison des algorithmes avant et après l'application de SMOTE :

	Random Forest		decision tree		XGBoost	
	0	1	0	1	0	1
Recall	92%	81%	85%	73%	87%	87%
Precision	88%	87%	84%	73%	92%	80%
F1-score	90%	84%	84%	73%	90%	84%
Moyene pondérée de rappel	88%		80%		87%	
Moyene pondérée de precision	8%		80%		87%	
Temps d'exécution	12m74s		9,3s		3m17s	
ROC AUC	93.88%		78.85%		93.09%	
Accuracy	87%		80%		87%	

FIGURE 10.1 – Comparaison des résultats avant de d'appliquer SMOTE

	Random Forest		decision tree		XGBoost	
	0	1	0	1	0	1
Recall	88%	93%	85%	84%	88%	93%
Precision	92%	87%	84%	85%	92%	88%
F1-score	90%	90%	85%	85%	90%	90%
Moyenne pondérée de rappel	90%		85%		90%	
Moyenne pondérée de précision	90%		85%		90%	
Temps d'exécution	12m74s		30.4s		3m17s	
ROC AUC	95.74%		84,53%		95.96%	
Accuracy	90%		85%		90%	

FIGURE 10.2 – Comparaison des résultats après l'application de SMOTE

### Interprétation

- Le ROC-AUC est plus faible sur décision tree , alors que Random forest et XGBoost sont proches (XGBOOST QUI a la meilleure valeur).
- Le temps d'exécution dans l'ordre croissant est toujours : Temps (Decision Tree) < Temps (XGBOOST) < Temps (Random Forest)
- Accuracy (Decision Tree) < Accuracy(Random Forest ) = Accuracy (XGBOOST)
- ROC-AUC( Décision tree ) < ROC-AUC( Random Forest ) < ROC-AUC( XG-BOOST).
- Parmi les prévisions positives prédites par le modèle XGBoost 88% sont réellement correctes (pour random forest, 87% sont correctes).

D'après ces résultats, le modèle optimal choisi est **XGBoost**.

Afin d'évaluer notre modèle sur deux ensembles de données correspondants à les années 2020 et 2021 et nous avons eu ces résultats :

	2020	2021
nb d'enregistrements	989 927	980 958
Score	89.21%	91.47%

FIGURE 10.3 – Comparaison des résultats pour les années 2020 et 2021

Parmi tous les classes négatives et positives 89% parmi elles sont prédit correctement pour l'année 2020 et 91% pour l'année 2021.

## 10.7 Déploiement du modèle

Pour faciliter la prédiction au membres de l'organisme d'accueil, nous avons créer un Dashboard en utilisant logiciel Open source **Power BI**, qui sert à afficher le résultat de prédiction en utilisant le modèle optimal sous formes des représentations graphiques.

Pour ce faire, nous allons à la surface de Power Bi , cliquant sur " obtenir des données d'une autre source".



FIGURE 10.4 – Surface d'accueil de Power Bi

On choisit *script python* afin de récupérer les données à visualiser

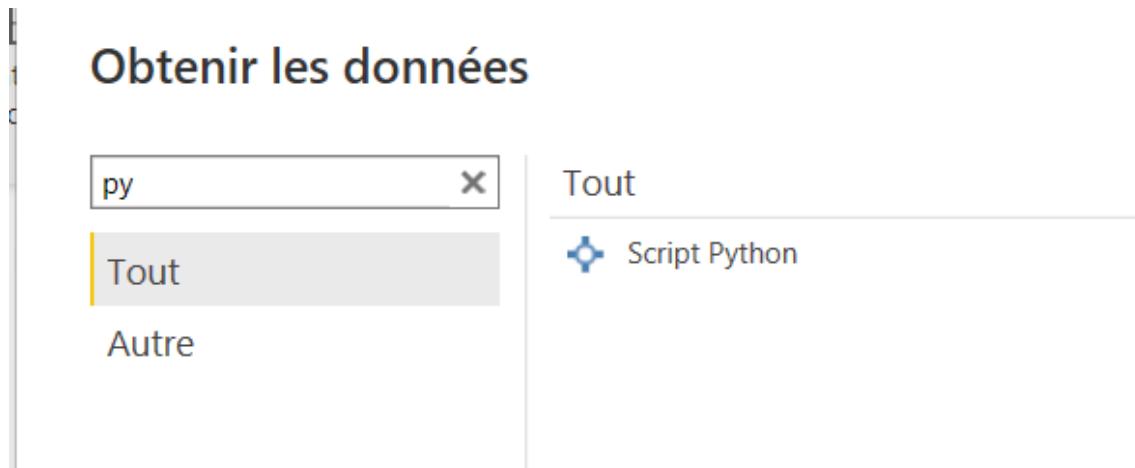


FIGURE 10.5 – Fenêtre de récupération de données

On clique sur *se connecter*, une fenêtre s'affichera, et on va saisir notre script python de notre modèle de Machine Learning.

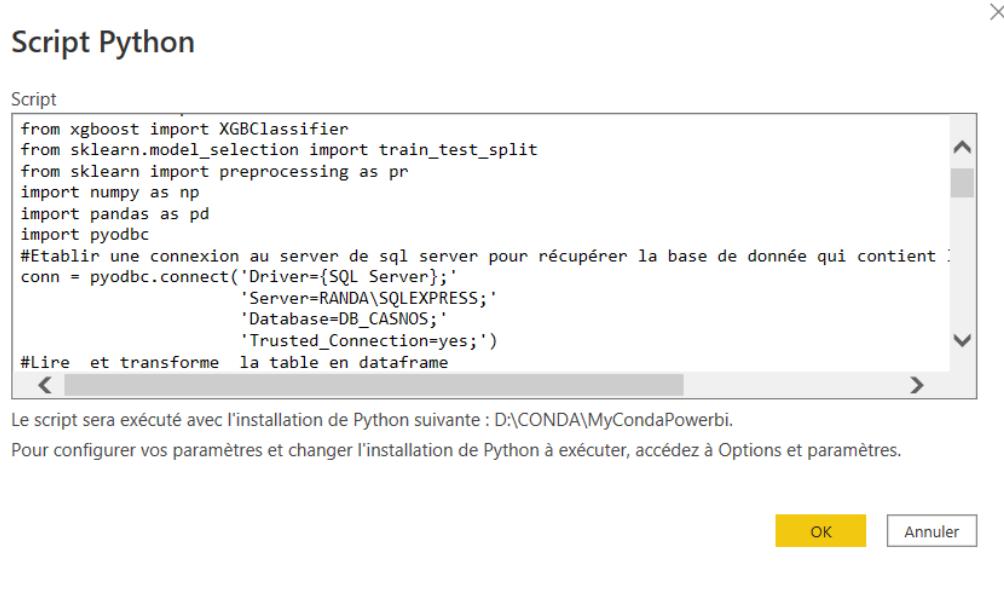


FIGURE 10.6 – Interface de script python

On clique sur OK, et on attend jusqu'à nos données sont importées et on commence par la suite par la réalisation de visualisations.

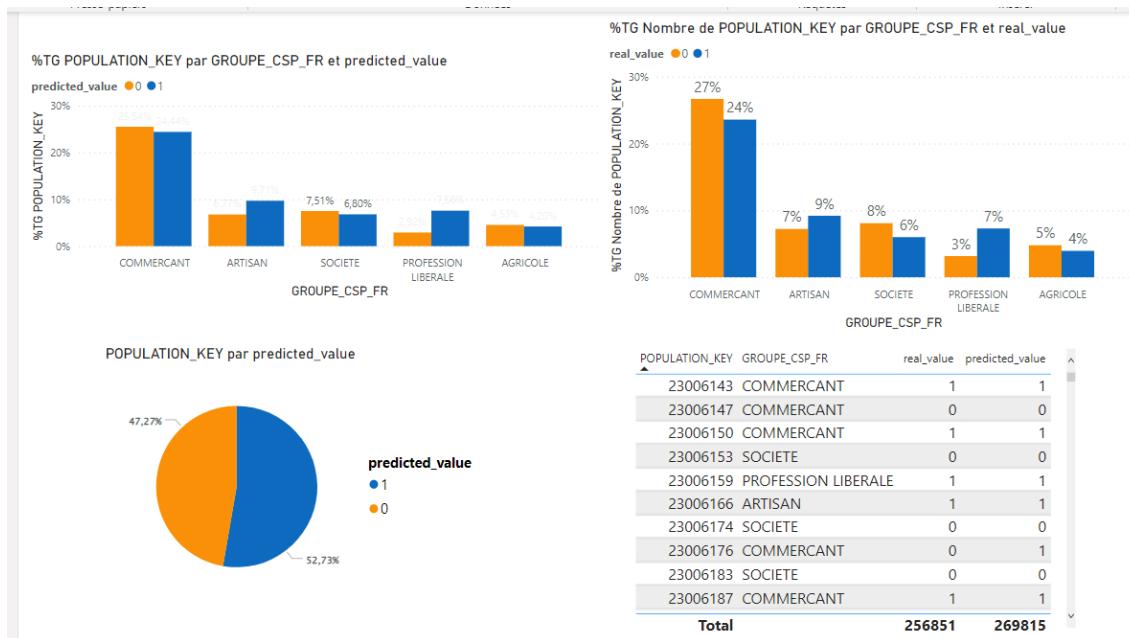


FIGURE 10.7 – Déploiement du modèle sous forme d'un dashboard sur Power Bi

## 10.8 Conclusion

Nous avons implémenté un modèle de machine learning pour la segmentation des clients tout en utilisant l'ensemble de données fournit par l'organisme CASNOS. Le développement avait de nombreux problèmes comme le déséquilibre et l'étiquetage des données. Afin de surmonter ces problèmes, nous avons appliqué la méthode de sur-

échantillonnage SMOTE avec des différents pourcentages qu'ont été essayé afin de surmonter le problème de déséquilibre de données, ensuite, de nombreuses expériences ont été faites pour trouver le meilleur algorithme avec les bons hyper-paramètres. Les résultats des modèles proposés étaient meilleurs que celles qui sont obtenues sans l'application de en utilisant les mêmes ensembles de données d'apprentissage et de tests.

Le modèle choisi est celle de XGBoost qui a donné des résultats très satisfaits après l'évaluer avec 2 jeu de données correspondants aux années 2020 et 2021 avec une précision très élevée.

Néanmoins,Les résultats obtenus sont très satisfaisants et le modèle choisi est très acceptable par rapport à la qualité des données utilisées.

# Chapitre 11

## Autres cas d'usages

### 11.1 Exécution des jobs spark sous SQL SERVER BIG Data Cluster

#### 11.1.1 Introduction

L'un des scénarios clés pour les clusters Big Data est la possibilité de soumettre des travaux Spark pour SQL Server. La fonctionnalité de soumission de tâches Spark vous permet de soumettre des fichiers Jar ou Py locaux avec des références au cluster Big Data SQL Server 2019. Dans ce chapitre nous allons, exécuter des jobs spark au seins de notre plateforme SQL SERVER BIG DATA sur une instance sql server(DBCASNOS).

#### 11.1.2 Soumettre des travaux Spark sur le cluster Big Data SQL Server dans Azure Data Studio

Nous avons suivit les etapes suivantes :

1. Ouvrir la boîte de dialogue de soumission de tâche Spark.

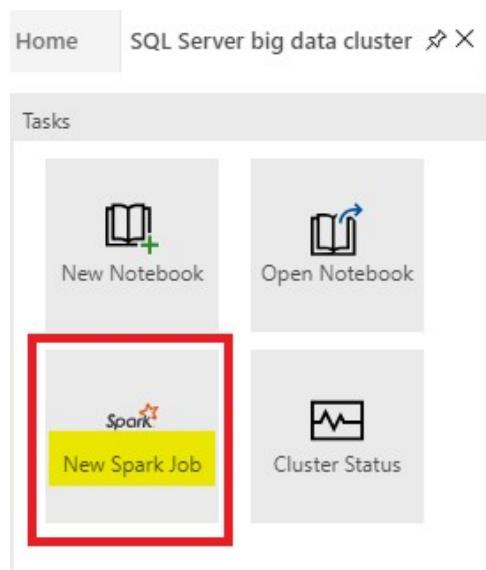


FIGURE 11.1 – Nouveau Spark Job

## Chapitre 11. Autres cas d'usages

---

2. La boîte de dialogue de soumission de tâche Spark s'affiche comme suit. Entrer le nom du travail, le chemin du fichier JAR/Py, la classe principale et d'autres champs.

The screenshot shows the 'New Job' dialog box with the 'GENERAL' tab selected. The 'Job Name' field contains 'Casnos spark test 1'. The 'Spark Cluster' field contains 'https://10.50.1.188:30443'. The 'JAR/py File' dropdown is set to 'Local' and the path '/home/srv-k8s-master-01/spark/casnos test spark1.py' is selected. A note below says 'The selected local file will be uploaded to HDFS: /SparkSubmission/2022/9/10/D7356EAF-7550-4345-8D1E-1D10AFC1B5E3/casnos test spark1.py'. The 'Main Class' and 'Arguments' fields are empty.

FIGURE 11.2 – Boîte de dialogue de soumission de Job Spark

3. Une fois la tâche Spark soumise, les informations sur la soumission et l'état d'exécution de la tâche Spark s'affichent dans l'historique des tâches sur la gauche. Les détails sur la progression et les journaux sont également affichés dans le OUTPUT en bas.

```
..... Submit Spark Job Start .....
[Info] Uploading file from local /home/srv-k8s-master-01/spark/casnos_spark_test.py to HDFS folder: /SparkSubmission/2022
[Info] Upload file to cluster Succeeded!
[Info] Submitting job casnos spark test ...
[Info] The Spark Job has been submitted.
[Info] Spark History Url: https://10.50.1.188:30443/gateway/default/sparkhistory/history/application_1662651449443_0003/1
[Info] YarnUI Url: https://10.50.1.188:30443/gateway/default/yarn/cluster/app/application_1662651449443_0003
..... Submit Spark Job End .....
```

FIGURE 11.3 – Surveiller la soumission des tâches Spark

4. Nous pouvons accéder à l'interface Spark via le lien fournit par l'action de soumission.

The screenshot shows the Apache Spark 3.3.0 UI interface. At the top, there is a header bar with back, forward, and search icons, and the URL 10.50.1.184:4042/jobs/. Below the header is a navigation bar with tabs for Jobs, Stages, Storage, Environment, and Executors. The main content area is titled "Spark Jobs (?)". It displays user information (User: root), total uptime (Total Uptime: 31 s), and scheduling mode (Scheduling Mode: FIFO). There is also a link to "Event Timeline".

FIGURE 11.4 – UI Spark job

## 11.2 La virtualisation des données

En générale , les données qui circulent au sein de la CASNOS sont des données structurées(SQL SERVER) mais il existe des cas où les informaticien ont besoin d'intégrer des données non structurées (NOSQL) par exemple .

### 11.2.1 Cas d'utilisation Des Données Non Structurées :

**Structure du nouveau système intégrer de la CASNOS** Une base de données NoSql MongoDB est utilisé dans ce nouveau système, elle permet de gérer les :

1. Logs : Afin de traiter les demandes d'entrée/sortie d'appelle API vers notre système et de détecter les différentes Bugs remonter depuis les agences.
2. Stockage de Fichier : Afin de stocker les fichiers de différentes types (pdf, excel, word, image scanner... etc) et ça nous permet une visualisation rapide des documents.

Notre plateforme Big Data offre la possibilités d'interroger des sources de données externes sans déplacer ni copier les données. En tirant parti de PolyBase.

#### 11.2.1.1 Étapes A suivre afin d'intégrer des données NOSQL (MongoDB)

1. Appuyer sur "Virtualize Data"

The screenshot shows a browser interface for data virtualization. At the top, there are buttons for "New Query", "New Notebook", "Restore", "Virtualize Data" (which is highlighted in yellow), "Refresh", and "Learn More". Below the buttons, it shows "Version : 15.0.4178.15" and "Computer Name : master-0". On the right, it shows "Edition : Developer Edition (64-bit)" and "OS Version : Ubuntu 20.04".

FIGURE 11.5 – Data virtualization

2. Choisir le type de données à virtualiser
3. Puis créer une connexion externe avec la source de données

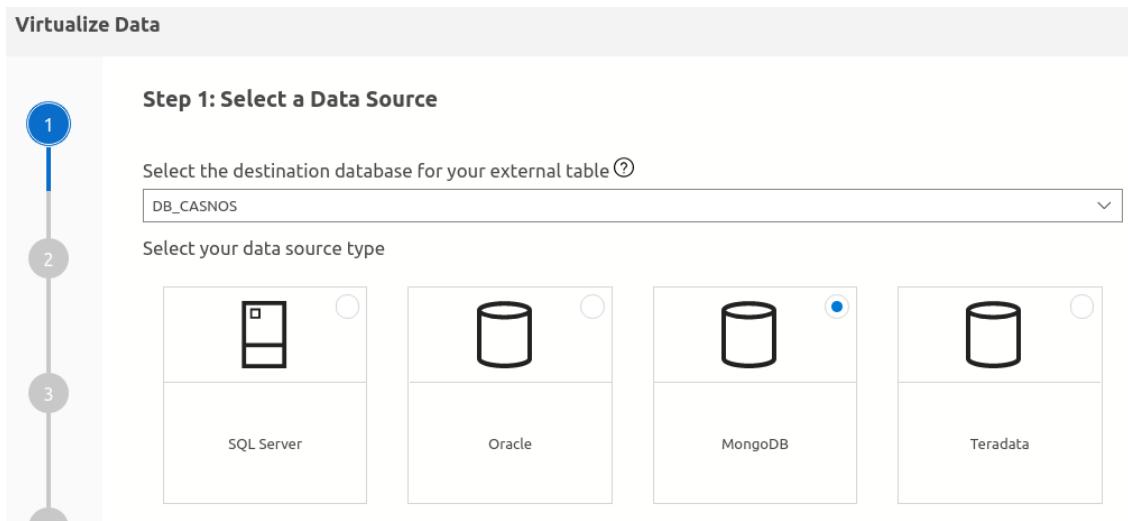


FIGURE 11.6 – Choisir le type de données à virtualiser

FIGURE 11.7 – Crédit de Connexion Externe Avec Source De Données

### 11.3 Exploration, visualisation et découverte des données logs avec Kibana

La plateforme SQL SERVER BIG DATA CLUSTER nous donne la possibilité d'avoir les journaux relatifs au cluster Big Data stockés dans Elasticsearch incluent les journaux de sortie et d'erreur standard de tous les services, notamment SQL Server, Spark, HDFS et les services de plateforme en temps réel ce qui peut être utiles au futur pour des raisons d'analyse etc . La figure ci-dessous illustre la dashboard des logs d'un service SPARK .

## Chapitre 11. Autres cas d'usages

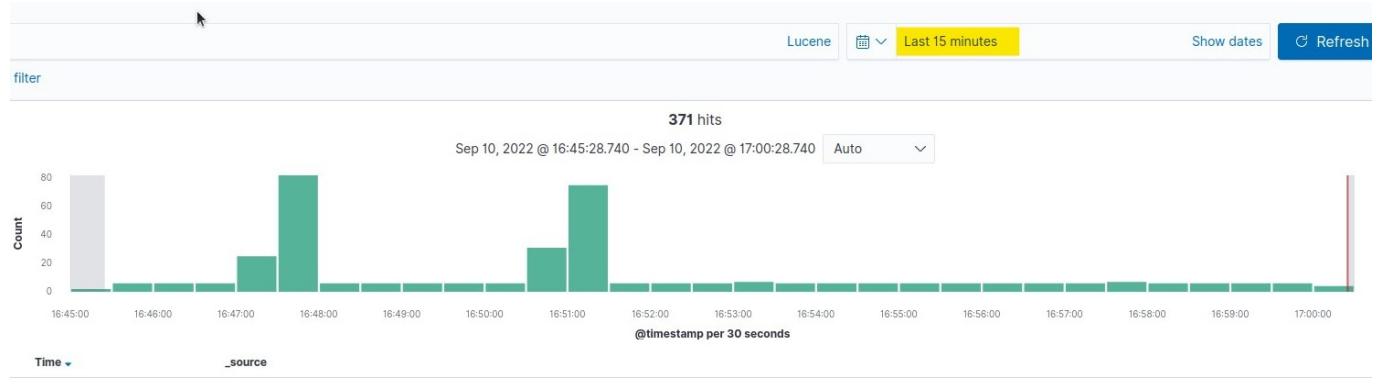


FIGURE 11.8 – Log Spark Avec Kibana Et ElasticSearch

## 11.4 Conclusion

Dans ce chapitre nous avons proposer quelque cas d'usage de la plateforme BIG DATA comme la machine learning, exécution des Jobs sparks, virtualisation de données avec le Polybase.

## **Sixième partie**

# **Conclusion Et Perspectives**

---

## 11.5 Conclusion générale

Au cours des derniers mois, nous avons travaillé sur un projet d'envergure où nous avons pu exploiter certains acquis de notre formation master et les mettre en pratique dans un milieu professionnel. Ainsi, la CASNOS nous a offert l'opportunité de réfléchir à une problématique importante pour l'entreprise et de contribuer à la résoudre et l'améliorer.

L'objectif initiale de ce PFE est l'amélioration la qualité de services vis-à-vis de ses usagers (Adhérents, assurés sociaux et retraités) via alignement un écosystème Big Data qui répond aux besoins de la société basé sur SQL SERVER BIG DATA CLUSTER sur un Cluster Kubernetes. Ce dernier offre divers services intégrés tel la visualisation de données real-time. En outre la plateforme a comme objectif d'améliorer le temps de réponse des requêtes complexes de plus, afin de faire face aux besoins d'évolution de la caisse le système se doit d'être flexible, évolutif et scalable .Chose qui est assurer à l'aide de Kubernetes.

ce mémoire est structurée de sorte à donner au lecteur une vision sur le cheminement de notre travail, ayant permis d'aboutir à la phase de réalisation de la solution. Pour ce faire, le travail a commencé par une collecte progressive de connaissances théoriques et pratiques dans les domaines de Big Data et Kubernetes. Ensuite on s'est prolongés dans l'aspect métier afin de collecter et identifier les différents besoins de l'organisme d'accueil. Une fois le cadre du projet et les besoins fixés, la conception et le déploiement de notre solution ont été effectués, en évaluant également pour démontrer sa qualité et ses performances par rapport à des solutions traditionnelles .

Ce stage nous a offert une véritable immersion dans le monde professionnel et les nombreux défis quotidiens rencontrés dans le milieu, et nous avons pu acquérir de nombreuses connaissances et une expérience valorisante en ce sens. Enfin, nous espérons que notre contribution apporte des idées originales et des possibilités d'évolution future, en offrant notamment un aperçu d'une solution envisageable et exploitable pour la CASNOS .

### 11.5.1 Présentation Du Future Système Informatique De La CASNOS Base Sur SQL SERVER Big Data

Le diagramme suivant illustre un aperçu sur le future système de la CASNOS base sur notre plate forme SQL SERVER Big Data . En principe , le système peut accueillir des données structurées(SQL Server) et non structurées (MongoDB) via le polybase ainsi que la possibilité d'ingestion de données(logs , CDC) via le biais d'un cluster NIFI et KAFKA .

Dans le cadre d'urbanisation, la plate forme offre aussi la possibilités de les stockées dans le Data lake ou les utilisées comme données d'entrée pour des requêtes SQL ou Spark .Pour enfin tirer de la valeur (Reporting ,BI , DWH , Machine learning etc).Toutes ces fonctionnalités circulent dans un système robuste,scalable et de haute disponibilité.

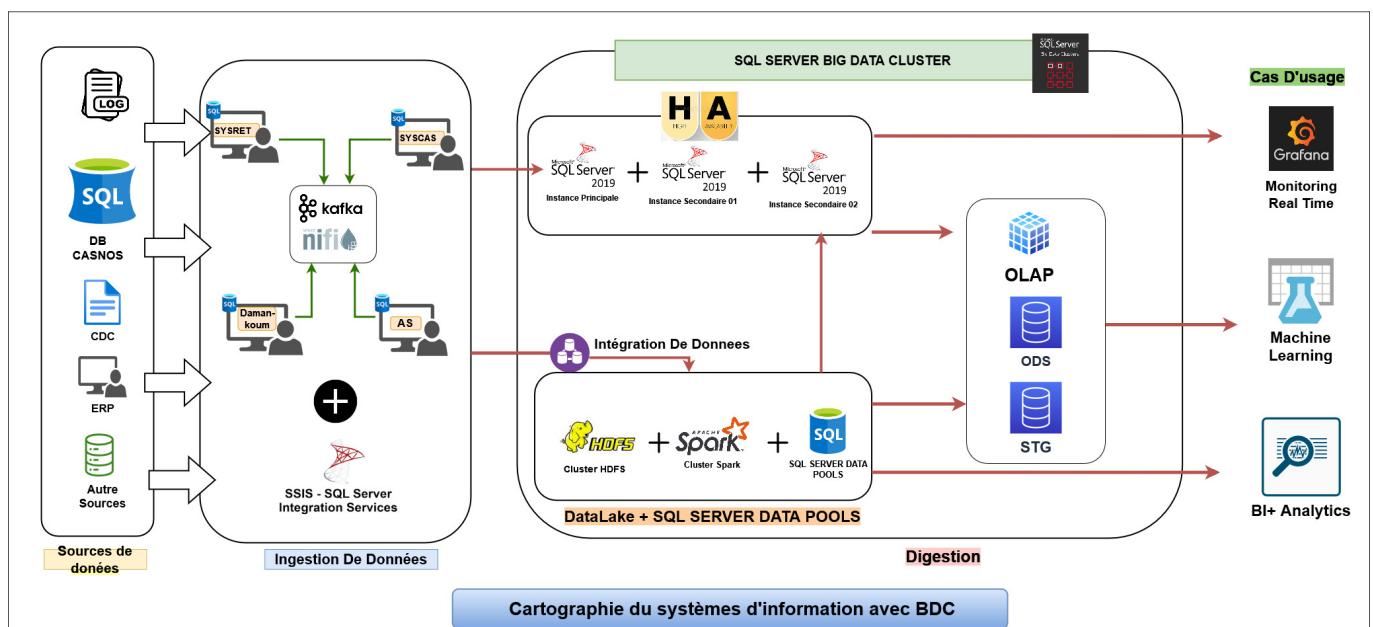


FIGURE 11.9 – Future Système Informatique De La CASNOS Base sur SQL SERVER Big Data

## 11.6 Perspectives

Le but initial de ce projet de fin de cycle est de contribuer à la concrétisation de la stratégie de réalisation du plan d'action 2021-2026 de la CASNOS via la mise en place une plateforme SQL SERVER BIG DATA CLUSTER robuste qui pourra accueillir des services au future Nous pouvons citer :

### 11.6.1 Déploiement Du Système D'information Et De Gestion D'assurance :OPENIMIS :

C'est un outil open source pour la gestion et l'organisation des processus métier liées à la sécurité sociale . Permet de numériser le lien entre les bénéficiaires, les prestataires

---

et payeurs ou tous ceux qui fournissent les fonds pour les programmes de sécurité sociale. Il est entouré de communauté de développeurs, d'utilisateurs et d'acteurs.



FIGURE 11.10 – Logo OpenIMIS

#### 11.6.1.1 Avantages OpenIMIS



##### **Plate-forme gratuite en accès libre**

Construisez sur des bases solides. Économisez des ressources. Profitez de ce que d'autres ont développé.

##### **Traitement automatisé des demandes**

Réduisez le travail administratif en reliant les données des patient·e·s aux prestataires de soins de santé.

##### **Dématérialisation des documents administratifs**

Diminuez les coûts opérationnels et augmentez l'efficience. Éliminez les tableaux Excel encombrants ou les factures papier.

##### **Amélioration de la gestion des données**

Utilisez les données pour améliorer la gestion des régimes, la prise de décisions politiques et les prestations en matière de soins de santé.

FIGURE 11.11 – Avantages OpenIMIS

#### 11.6.2 Envoi SMS en Temps Réel pour les usagers de la CARTE CHIFA

#### 11.6.3 L'ingestion de Donnes Logs et CDC à l'aide d'un cluster NIFI ET KAFKA

# Références

- [1] *Introduction à l'apprentissage automatique.* Visité le : 19.08.2022. DOI : [https://projeduc.github.io/intro\\_apprentissage\\_automatique/regression.html](https://projeduc.github.io/intro_apprentissage_automatique/regression.html).
- [2] *La sécurité sociale Et financement du système de soins en Algérie.* Visité le : 03.08.2022. DOI : [https://fmed.univ-tlemcen.dz/ressources/documents\\_actualite/scolimed\\_160.pdf](https://fmed.univ-tlemcen.dz/ressources/documents_actualite/scolimed_160.pdf).
- [3] *Le clustering, définition et implémentation.* Visité le : 19.08.2022. DOI : <https://analyticsinsights.io/le-clustering-definition-et-implementations/>.
- [4] *Matrice de confusion, la comprendre et l'utiliser.* Visité le : 23.08.2022. DOI : <https://kobia.fr/classification-metrics-matrice-de-confusion/>.
- [5] Zakia MESSAOUDI. *Les 3 étapes essentielles de l'apprentissage automatique (Machine Learning).* Visité le : 18.08.2022. 22 janvier 2020. DOI : <https://www.spiria.com/fr/blogue/intelligence-artificielle/3-etapes-essentielles-apprentissage-automatique-machine-learning>.
- [6] Jean-Yves RAMEL. « De l'analyse de données et l'apprentissage automatique ». In : () .
- [7] *Régression et Classification / Apprentissage automatique supervisé.* Visité le : 18.08.2022. DOI : <https://fr.acervolima.com/regression-et-classification-apprentissage-automatique-supervise/>.

## Annexes

# Annexe A

## Préparation De L'environnement De Travail

### A.1 Introduction

Ce chapitre est dédié à la partie de préparation de l'environnement de travail. Ce sont les étapes à faire dans partie pré-déploiement.

### A.2 Préparation Du Serveur

C'est l'équipe système de la CASNOS qui s'en charge de préparer le serveur : installation de l'OS (Windows Server 2019) , mises à jours ainsi que l'établissement de connections réseaux . En plus,l'équipe doit installer un rôle Hyper-V , le configurer en passant par les étapes suivantes :

1. La création du réseau virtuel (Commutateur virtuel)
2. Migration dynamique
3. Emplacement par défaut des ordinateurs virtuels

Afin d'accéder à l'hyperviseur :Taper "Hyper-V" dans la barre de recherche. Après avoir



FIGURE A.1 – Gestionnaire Hyper-V

installer un rôle Hyper-V , nous pouvons le trouvez dans le menu principale .

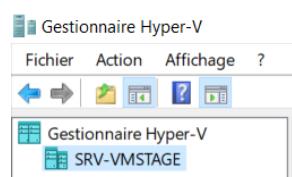


FIGURE A.2 – Serveur Hyper-V

## A.3 Création Des Machines Virtuelle via Hyper-V

Les instructions pour la création des Vms que ce soit serveur Master ou esclave (Worker) sont les mêmes la différence est dans les capacité :ROM/RAM. Afin de créer des VMs sur Hyper-V nous avons suivi les étapes suivantes :

1. Cliquer sur le nom de la machine Hôte afin de la sélectionner. Aller ensuite dans le menu Action pour cliquer sur "Nouveau" puis sur "Ordinateur virtuel"

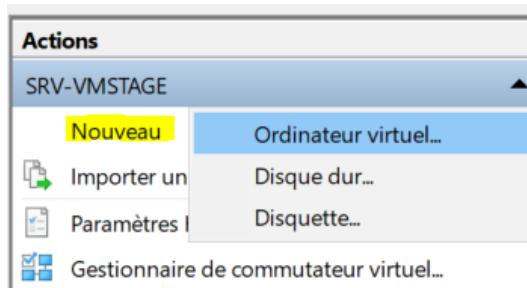


FIGURE A.3 – Nouveau Ordinateur Virtuel Hyper-V

2. L'assistant Nouvelle machine virtuelle s'ouvrira, présentant l'ensemble des options de machine virtuelle que nous devons configurer.
3. Dans la section suivante, nous pouvons configurer le nom et l'emplacement de la machine virtuelle. Nous avons choisi "**SRV-K8S-MASTER-01**" = Serveur Kubernetes Master01



FIGURE A.4 – Nommer la VM

4. Dans la section Génération, de la VM. Le choix entre la Génération 1 et la Génération 2 est principalement dicté par l'OS invité que nous souhaitons installer (64/32 bits) .

- Ensuite, nous devons spécifier la quantité de mémoire (16G dans notre cas) qui sera attribuée à la VM. Les performances futures de la VM dépendront largement de la quantité de mémoire allouée.

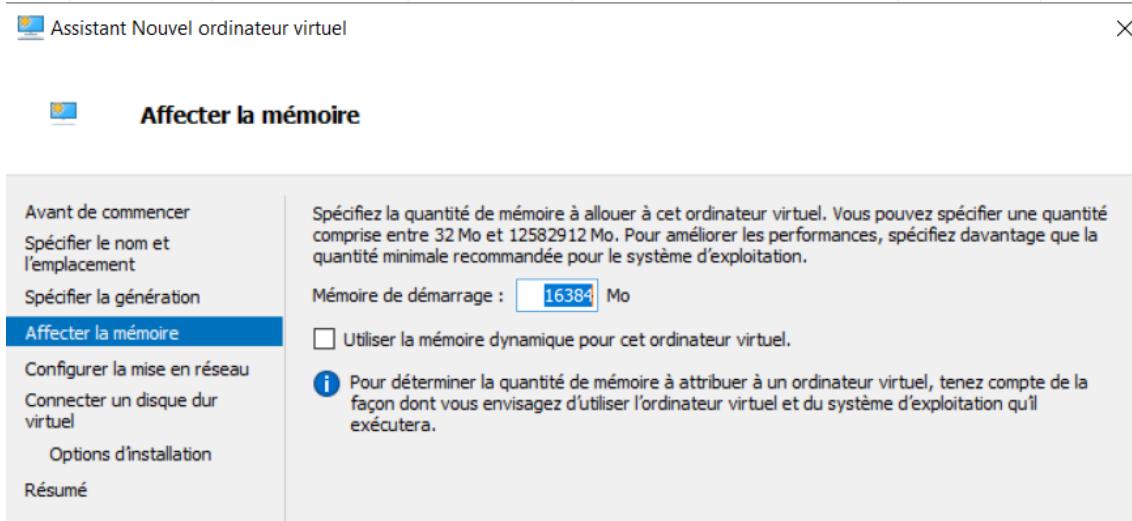


FIGURE A.5 – Spécifier la quantité de mémoire VM

- L'étape suivante permet de choisir un réseau virtuel qui sera utilisé pour connecter la VM au réseau. Pour cela, nous avons sélectionné un commutateur virtuel déjà créé précédemment par l'équipe Système .

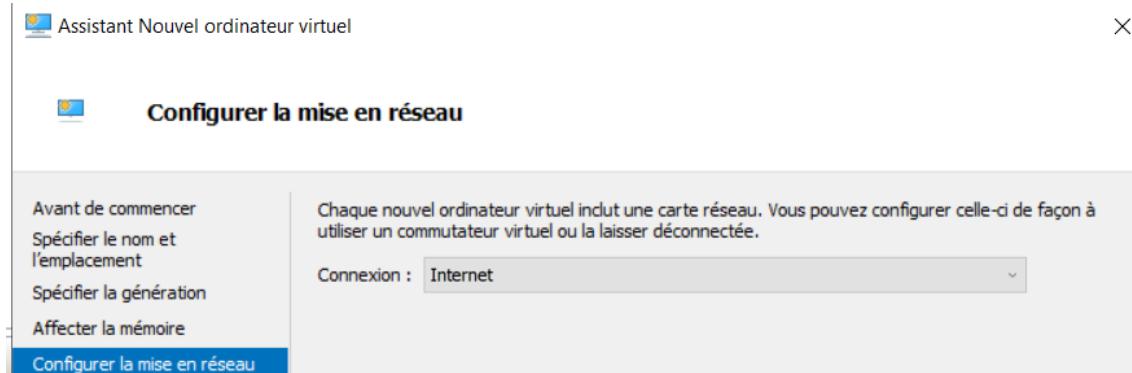


FIGURE A.6 – Choisir un réseau virtuel VM

- Ensuite configurer les exigences du disque dur virtuel. Dans cette section, nous pouvons créer un nouveau disque dur virtuel, ce qui nécessite de spécifier son nom, son emplacement et sa taille. Vu que nous avons types de Vms :Master Worker et les nodes workers nécessite plus de mémoire de stockage que le Master donc la distribution est la suivantes :
  - Master Node : 170 G
  - Worker Node : 230 G

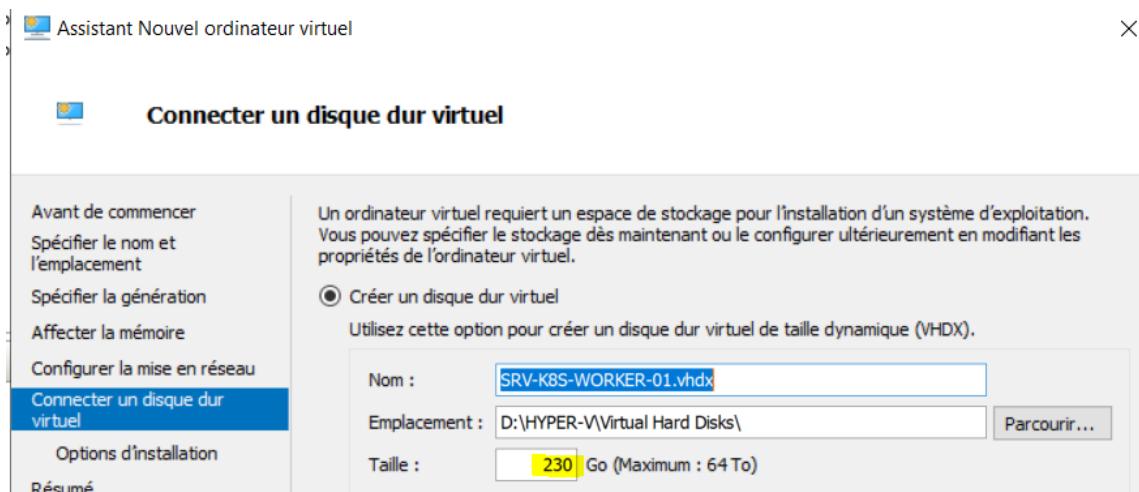


FIGURE A.7 – Choisir le disque dur virtuel de la VM

- La section Options d'installation apparaîtra où nous pouvons spécifier les configurations :système d'exploitation invité en choisissant le fichier ISO. Dans notre cas , nous avons choisi un fichier ISO contenant une image Ubuntu 20.04.

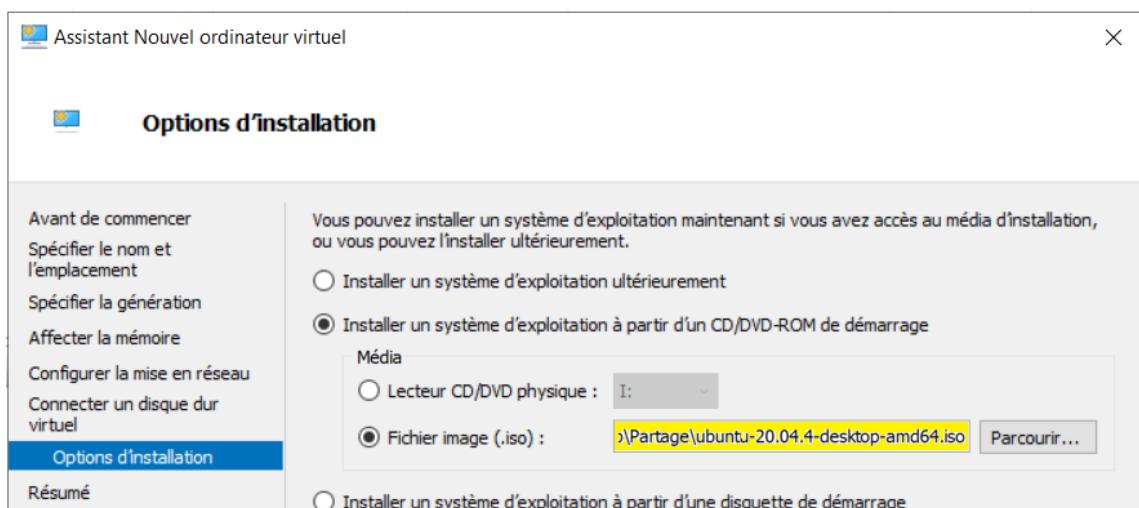


FIGURE A.8 – Choisir Système D'exploitation Invité de la VM

- La dernière section est Résumé, qui fournit une brève description de cette VM.le parcourir à nouveau et vérifiez que tout est correct. Si tel est le cas, cliquer sur Terminer pour créer la machine virtuelle et fermer l'assistant.
- Après avoir effectuer la configuration et pour lancer l'installation de la Vm , cliquer sur la Vm => Se connecter

Ordinateurs virtuels						
Nom	État	Utilisation d...	Mémoire affectée	Temps d'activité	Statut	Version de c...
SRV-K8S-MASTER-01	En cours	0 %	16884 Mo	00:00:58		9.0

FIGURE A.9 – Se connecter à la VM

11. Par la suite, lancer l'installation d'Ubuntu

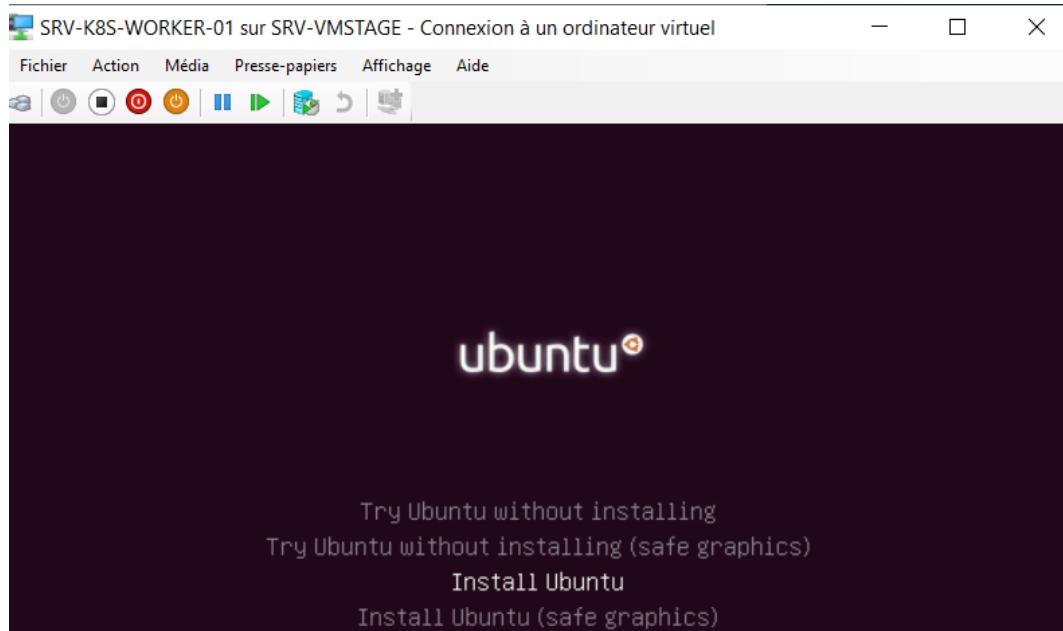


FIGURE A.10 – Lancer l'installation Ubuntu à la VM

## A.4 Conclusion

Nous avons lister les différentes étapes à suivre afin de créer une VM sur Hyper-V . Dans le prochain chapitre , nous allons déployer un cluster Kubernetes .

## Annexe B

# Choix Technologique

### B.1 Introduction

Ces dernières années, nous avons vu beaucoup de discussions sur l'avalanche de données au sein de l'écosystème numérique. La révolution numérique a progressé dans une réaction en chaîne, à commencer par l'invention de l'information, le développement d'Internet et l'essor des smartphones. Le cycle de l'innovation se poursuit et continue de s'accélérer. Ainsi, la technologie du big data contribue également à l'émergence de nouveaux processus, applications et habitudes, influençant fortement notre vie quotidienne et professionnelle.

Cependant, le choix des outils et technologies d'application n'est pas toujours évident. Les entreprises testent toutes les technologies avant d'entamer la réalisation du projet via POC\* (preuve de concept)<sup>1</sup> et découvrent que certaines ne sont pas applicables dans des situations réelles.

Il y a plusieurs raisons à cela. Les technologies peuvent être trop futuristes, peu personnalisables, incompatibles avec les grands systèmes d'information internes, complexes à utiliser par les intervenants sur le terrain, voire trop coûteuses.



FIGURE B.1 – La Transformation Digitale

1. Une preuve de concept ou validation de principe, ou encore démonstration de faisabilité, est une réalisation ayant pour vocation de montrer la faisabilité d'un procédé ou d'une innovation

## B.2 Comment choisir la bonne technologie pour conduire une véritable transformation ?

Il faut prendre en considération les critères suivants :

- La valeur ajoutée
- Rapport bénéfices/coûts
- La compatibilité avec les systèmes d'information déjà existant
- La capacité à être déployée
- L'universalité de la solution

## B.3 Les Technologies utilisées

### B.3.1 Sql Server 2019

Microsoft SQL Server est un système de gestion de base de données (SGBD) en langage SQL qui comprend, entre autres, un RDBMS (SGBD relationnel) développé et commercialisé par Microsoft. Il fonctionne sur Windows et Linux OS (depuis mars 2016), mais il est également possible de le lancer sur Mac OS via Docker, car il existe une version téléchargeable sur le site de Microsoft.



FIGURE B.2 – Logo SQL Server 2019

### B.3.2 Sql Server Big Data Cluster

Cette fonctionnalité d'SQL SERVER 2019 consistant à rassembler toutes ces technologies et architectures dans la plate-forme de données, Microsoft a appelé Big Data Clusters (BDC) qui sont intégrés à SQL Server.

Les clusters Big Data SQL Server offrent une flexibilité lors de l'utilisation de Big Data. Avec la possibilité d'interroger des sources de données externes, stocker des mégadonnées dans des systèmes HDFS gérés par SQL Server et interroger des données provenant de plusieurs sources de données externes dans cluster. Les données peuvent ensuite être utilisées pour l'intelligence artificielle, l'apprentissage automatique et d'autres tâches analytiques.



FIGURE B.3 – Logo SQL Server Big Data Cluster

### B.3.2.1 Architecture SQL SERVER Big Data Cluster

Vu que SQL SERVER BDC est l'un des piliers de notre projet nous allons détailler son architecture si dessous . Microsoft avec BDC vise à mettre à la disposition de tous ses utilisateurs et clients une plateforme évolutive, hautement performante, sécurisée et robuste intégrant les technologies SQL, Data Warehouse, Data Lake et Data Science. BDC est disponible à la fois en « cloud » et « sur site ».

Voici une carte de l'écosystème du BDC :

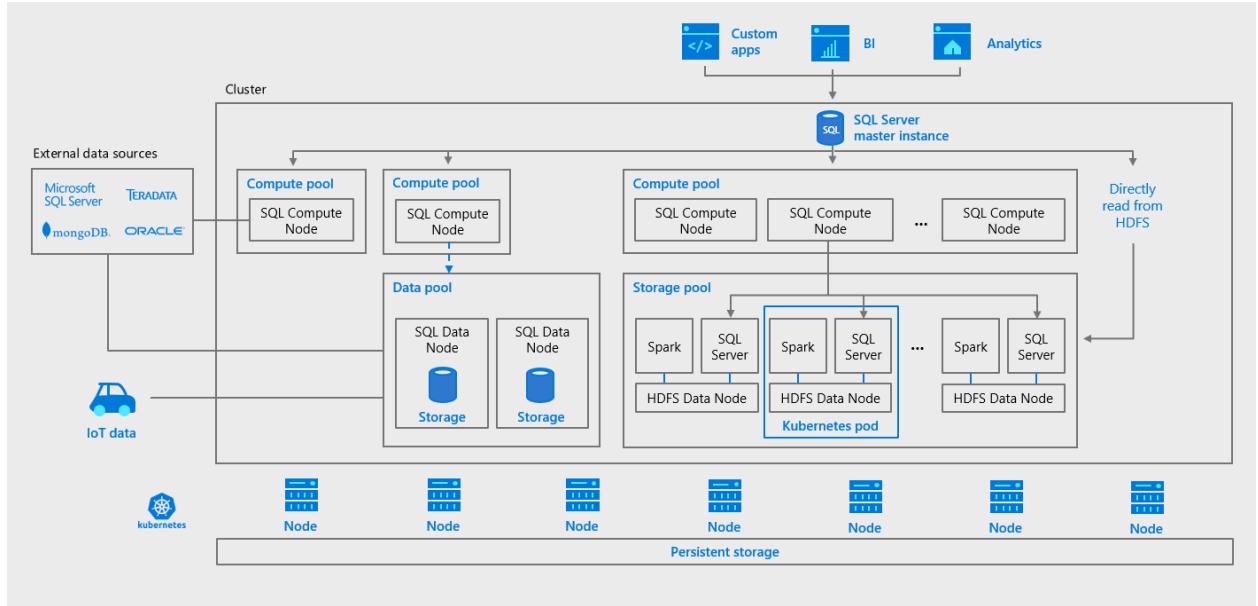


FIGURE B.4 – Architecture SQL Server Big Data Cluster

1. **Pool De Contrôleur** : Les contrôleurs assurent la gestion et la sécurité du cluster. Cela inclut le service de contrôle, le magasin de configuration et d'autres services au niveau du cluster tels que Kibana, Grafana et ElasticSearch.
2. **Pool de calcul** : Un pool de calcul fournit des ressources de calcul à un cluster. Contient des nœuds exécutant des pods SQL Server sur Linux. Les pods de pool de calcul sont divisés en instances de calcul SQL pour des tâches de traitement spécifiques.
3. **Pool de stockage** :  
Un pool de stockage se compose de pods SQL Server sur Linux, Spark et HDFS. Tous les nœuds de stockage d'un cluster SQL Server Big Data sont membres du cluster HDFS.
4. **Pool d'applications** : Les applications sont déployées sur les clusters Big Data SQL Server via une interface conçue pour créer, gérer et exécuter des applications.

### B.3.2.2 Technologies mises en œuvre par le BDC

- **Instance SQL Server** : Lorsque nous avons déployé l'instance SQL Server 2019 dans Kubernetes, nous avons dans notre architecture une instance "maître" et plusieurs moteurs SQL Server pour effectuer des opérations de calcul et de "sharding".

L'instance "maître" se comportera comme une instance SQL Server 2019. Lorsque nous devons diffuser des données vers notre cluster, nous pouvons directement utiliser nos instances sur des noeuds "non maîtres" ("shards"). Cela permet une amélioration des performances.

- **HDFS et Lac de données :** Lors de l'installation et de la configuration de SQL Server 2019 BDC, un système de fichiers distribué Hadoop (HDFS) est également installé dans Kubernetes. En utilisant les groupes "scale-out" de Polybase, nous pouvons accéder facilement et efficacement à ces données distribuées au sein de SQL Server vers des tables externes.

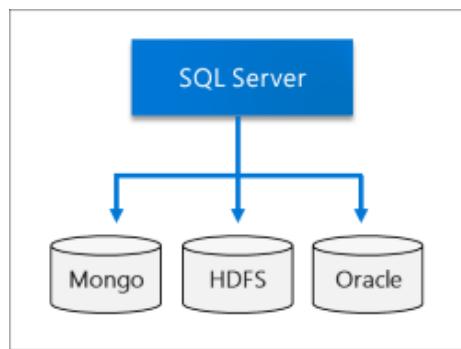


FIGURE B.5 – Date Lake dans SQL Server Big Data Cluster

- **Polybase :** Polybase était une fonctionnalité introduite dans le moteur de base de données SQL Server 2016, qui nous permettait de nous connecter aux sources de données HDFS. Avec SQL Server 2019, nous pouvons également nous connecter à des sources de données relationnelles (Oracle, SAP Hana, PostgreSQL, ...) ou à des sources de données NoSQL (MongoDB, Redis, Apache Cassandra, Azure Cosmos DB, ...). Nous pouvons même utiliser Polybase avec des tables externes pour atteindre nos objectifs et être plus productifs en traitant toutes les données plus rapidement.

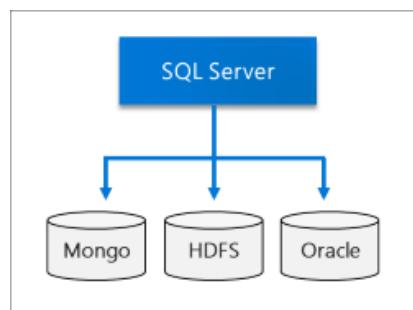


FIGURE B.6 – Polybase dans SQL Server Big Data Cluster

#### — Apache Spark :

Pour ceux qui ont l'habitude de travailler avec des projets et applications Spark, il ya la possibilité de profiter de toutes les fonctionnalités (SparkSQL, Dataframes, MLlib, ...) au sein de cluster SQL. Dans l'organisation, avec l'équipe Data Science et Data Engineers, il est possible de faire de SQL Server 2019 BDC un point de centre de données « Big Data ».

- **IA et Machine Learning intégrés** Les clusters Big Data SQL Server permettent des tâches d'IA et d'apprentissage automatique sur les données stockées dans des pools de stockage et des pools de données HDFS. En outre, il permet d'utiliser les outils Spark et AI intégrés à SQL Server à l'aide de R, Python, Scala ou Java .

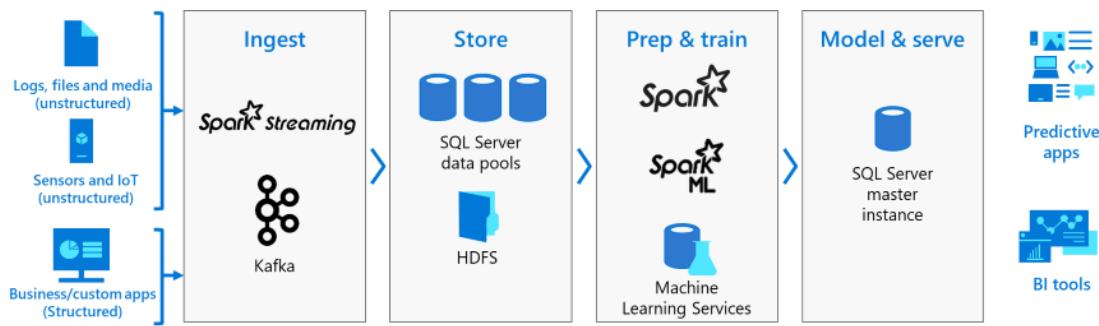


FIGURE B.7 – AI/ML dans SQL Server Big Data Cluster

#### B.3.2.3 Pourquoi SQL SERVER BDC ?

Utiliser les clusters Big Data SQL Server pour :

- Déployer des clusters scalables : de conteneurs SQL Server, Spark et HDFS exécutés sur Kubernetes.
- Lire, écrire et traiter les données du Big Data à partir de Transact-SQL ou de Spark.
- Combiner et analyser facilement des données relationnelles à valeur élevée et un volume important de données du Big Data.
- Interroger des sources de données externes.
- Stocker les données du Big Data dans un système HDFS géré par SQL Server.
- Interroger les données de plusieurs sources de données externes via le cluster.
- Utiliser les données pour l'IA, le machine learning et d'autres tâches d'analyse.
- Déployer et exécuter des applications dans les Clusters Big Data.
- Virtualiser les données avec Polybase.
- Interroger les données de sources de données SQL Server, Oracle, Teradata, MongoDB et ODBC génériques avec des tables externes.
- Fournir la haute disponibilité pour l'instance principale SQL Server et toutes les bases de données à l'aide de la technologie des groupes de disponibilité Always On.

#### B.3.2.4 Valeurs Ajoutées Pour La CASNOS :

Les raisons principales du choix de SQL SERVER pour la réalisation de la solution Big Data sont les suivantes :

1. **Solution Économique** : La CASNOS elle bénéficie d'un abonnement SQL SERVER 2019. D'où penser à une solution Big data basée sur des technologies déjà existantes au sein de l'entreprise est un point en plus sur le volet budgétaire.
2. **Facilité D'adaptation** : L'équipe technique de la DMSI à l'habitude de travailler avec SQL SERVER ainsi que notre solution .
3. **Nécessité De Changement** Les bases de données de la CASNOS augmentent de volume jour après jour et les solutions traditionnelles(Index, CDC ) peuvent ne pas suffire. Par conséquent il est grand temps de basculer vers la BIG DATA .
4. **Technologies Robustes** SQL SERVER BIG DATA repose sur de nouveaux concepts qui traitent des masses de données distribuées (HADOOP)en utilisant des bibliothèques puissantes(APACH SPARK).
5. **Plateforme Intégrée** SQL SERVER BDC offre une variété de possibilités d'interagir avec les données : analyse business intelligence ,dashboarding ou aussi pour des modèles Machine Learning .
6. **Concept De La Haute Disponibilité** L'architecture suggérée peut assurer la haute disponibilité informatique, par conséquent :
  - Gérer les pannes pour maintenir les activités en cours
  - Augmenter la rentabilité en minimisant l'impact financier des pannes.
  - Maintenir la confiance et la responsabilité des clients, partenaires et actionnaires.

### B.3.3 Apache HADOOP

Hadoop est un framework gratuit et open source écrit en Java qui vise à faciliter la création d'applications distribuées et évolutives, permettant aux applications de traiter des milliers de nœuds et des pétaoctets de données.



FIGURE B.8 – Logo Apach Hadoop

#### B.3.3.1 Valeurs Ajoutées Pour La Casnos

- **Coût** : Hadoop est open source qui offre un modèle à faible coût, contrairement aux bases de données relationnelles traditionnelles qui nécessitent du matériel coûteux et des processeurs haut de gamme pour traiter le Big Data.

- **Destiné à Big Data Analytics** : Hadoop est un concept de gestion du Big Data :Il peut gérer le volume, la variété, la vélocité et la valeur.
- **Diminue Temps De Latence** : Hadoop utilise un système de fichiers distribué pour gérer le stockage. DFS (Distributed File System) divise les fichiers volumineux en morceaux de fichiers plus petits et les distribue sur les nœuds disponibles dans un cluster Hadoop. En effet, ce grand nombre de morceaux de fichiers est traité en parallèle, ce qui accélère Hadoop.
- **Faisabilité** : Hadoop peut fonctionner sur du matériel de base, ce qui signifie qu'il ne nécessite pas de serveur très haut de gamme doté d'une mémoire et d'une puissance de traitement importantes. Hadoop s'exécute sur JBOD<sup>2</sup>, de sorte que chaque nœud est indépendant dans Hadoop.
- **Assure La Haute Disponibilité** : Dans Hadoop, les données sont répliquées sur divers DataNodes dans un cluster Hadoop, ce qui garantit la disponibilité des données en cas de panne de l'un des systèmes .
- **Haut débit** : Hadoop fonctionne sur un système de fichiers distribué, avec différentes tâches affectées à différents nœuds de données dans le cluster. Une grande partie de ces données est traitée en parallèle sur un cluster Hadoop, ce qui se traduit par un débit élevé.

### B.3.4 Apach Spark

Apache Spark est un moteur d'analyse unifié conçu pour le traitement de données à grande échelle, avec des modules intégrés pour SQL, le traitement de flux, l'apprentissage automatique et la création de graphiques. Spark s'exécute de manière autonome ou dans le cloud sur Apache Hadoop, Apache Mesos et Kubernetes. De plus, il peut être appliqué à diverses sources de données.



FIGURE B.9 – Logo Apach Spark

#### B.3.4.1 Valeurs Ajoutées Pour La Casnos

- **Rapidité de traitement de requêtes** La vitesse de traitement de données est élevée. Environ 100 fois plus rapide en mémoire et 10 fois plus rapide sur le disque. Ceci est rendu possible en réduisant le nombre de lecture-écriture sur le disque(les

---

2. Le JBOD est un système de stockage de données consistant à regrouper plusieurs disques durs en un seul volume. Il n'utilise ni la technologie de distribution de données, ni la technologie de la redondance. Littéralement, JBOD est l'acronyme de "Just a Bunch of Disks", juste un tas de disques.

données sont mises en cache)

- **Tolérance aux pannes** Les RDD Spark sont conçus pour gérer l'échec de tout nœud de travail du cluster. Ainsi, cela garantit une perte de données nulle.
- **Traitement de flux en temps réel** Spark fournit un traitement de flux en temps réel. Auparavant, le problème avec Hadoop MapReduce était qu'il pouvait traiter les données existantes et les traiter, mais il ne pouvait pas traiter les données en temps réel. Mais avec Spark Streaming, ce problème est résolu.
- **Convivialité pour les développeurs** Apache Spark prend nativement en charge Java, Scala, R et Python, fournissant une variété de langages pour la création d'applications. Ces API sont plus faciles pour les développeurs car elles cachent la complexité du traitement distribué derrière des opérateurs simples et de haut niveau qui réduisent considérablement la quantité de code requise.
- **Charges de travail multiples** Il offre la possibilité d'exécuter plusieurs charges de travail telles que des requêtes interactives, des analyses en temps réel, l'apprentissage automatique et le traitement de graphiques. Les applications peuvent combiner de manière transparente plusieurs charges de travail.
- **Spark s'intègre à l'architecture Hadoop** Spark peut fonctionner de manière autonome et en mode distribué car l'outil dispose de son propre mécanisme de clusterisation. Mais l'intérêt est de pouvoir bien entendu l'intégrer à un cluster Hadoop. Or, cela se fait très simplement. Yarn continue de centraliser les besoins et gérer le pilotage des ressources entre des traitements Spark et d'autres traitements de type Map Reduce.

### B.3.5 Docker

Docker est une plateforme de lancement d'applications spécifiques dans des conteneurs logiciels. Selon la société de recherche industrielle 451 Research, "Docker est un outil qui vous permet de regrouper des applications et leurs dépendances dans des conteneurs isolés pouvant s'exécuter sur n'importe quel serveur".



FIGURE B.10 – Logo Docker

#### B.3.5.1 Valeurs Ajoutées Pour La Casnos

- **Économies conséquentes** : Docker est open source. Par conséquent, il n'y a pas de frais de licence à l'achat. En plus, Les conteneurs Docker facilitent l'exécution de plus de code sur chaque serveur, améliorant ainsi l'expérience utilisateur et réduisant les coûts.

- **Notion de Micro-services** : Concevoir et mettre à l'échelle des architectures d'applications distribuées en tirant parti de déploiements de code standardisés à l'aide de conteneurs Docker.
- **Traitement de données Big Data** : Fournit un service de traitement de Big Data. Rassemble des données et des paquets d'analyses sous la forme de conteneurs portables qui peuvent être exécutés facilement.
- **Aide l'équipe technique**
  1. **Pour les développeurs** : Docker permet donc aux développeurs de s'assurer que leurs applications fonctionnent quel que soit le système d'exploitation ou l'environnement dans lequel elles sont publiées.  
Cela permet de se concentrer sur la production de code au lieu de passer du temps à réfléchir au système sur lequel application s'exécute.
  2. **Pour les administrateurs systèmes** : Docker permet d'installer et de démarrer des conteneurs qui fonctionnent ensemble. Combiné avec Docker-Compose, il est possible de déployer une application entière et ses dépendances avec une seule commande. Enfin, l'installation des mises à jour peut être simplifiée grâce à des configurations faciles à mettre en œuvre.
- **Tolérance Aux Pannes** : Ainsi, lorsqu'un conteneur Docker est hors service, contrairement aux technologies de virtualisation traditionnelles peuvent prendre beaucoup de temps à se redéployer et offrir des garanties insuffisantes de protection des données, mais la récupération et le redéploiement quasi instantanés sont très rapides.

### B.3.6 Kubernetes

Kubernetes est un système open source qui vise à fournir une plate-forme pour automatiser le déploiement, la mise à l'échelle et le déploiement de conteneurs d'applications sur des clusters de serveurs. Kubernetes regroupe des collections de conteneurs pour les gérer sur la même machine, ce qui réduit la surcharge du réseau et optimise l'utilisation des ressources. Des exemples d'ensembles de conteneurs sont les serveurs d'applications, les caches Redis et les bases de données SQL.



FIGURE B.11 – Logo Kubernetes

#### B.3.6.1 Valeurs Ajoutées Pour La Casnos

- **Solution Économique** Kubernetes est open source. Par conséquent, il n'y a pas de frais de licence à l'achat.

- **Déploiements et restaurations automatisés** Kubernetes est livré avec une API puissante et un outil de ligne de commande appelé kubectl qui effectue de nombreuses tâches de gestion de conteneurs qui permettent d'automatiser les opérations. Le modèle de contrôleur de Kubernetes garantit que les applications/conteneurs s'exécutent exactement comme spécifié.
- **Tolérance Aux Pannes et Auto-guérison** Redémarre les conteneurs qui échouent, remplace et replanifie les conteneurs lorsque les noeuds tombent, supprimer les conteneurs qui ne répondent pas aux vérification de l'état définie par l'utilisateur et ne les annonce pas aux clients tant qu'ils ne sont pas prêts à servir.
- **Mise à l'échelle horizontale** Augmenter et diminuer l'application avec une simple commande, avec une interface utilisateur ou automatiquement en fonction de l'utilisation du processeur
- **Adaptation Au Changement** Kube adapte et distribue la charge, mais il adapte également son infrastructure au fur et à mesure que les développeurs codent ses fonctionnalités, garantissant que tout fonctionne en harmonie.
- **La Disponibilité** Disponibilité sur site(on-permise) et dans le cloud sur divers plateformes tels que Google Cloud Platform, AWS, Microsoft Azure et IBM Cloud

#### B.3.6.2 Kubernetes VS Docker Swarm

Pour une comparaison significative, il est recommandé de comparer Kubernetes et Docker Swarm. Docker Swarm est un outil d'orchestration de conteneurs comme Kubernetes. Cela signifie qu'il peut gérer plusieurs conteneurs déployés sur plusieurs hôtes exécutant des serveurs Docker. Le mode Swarm est désactivé par défaut sur Docker et doit être installé et configuré par équipe DevOps.



FIGURE B.12 – Docker Swarm VS Kubernetes

Ci dessous un tableau comparatif entre Kubernetes et Docker Swarm

Kubernetes	Docker Swarm
Developed by Google	Developed by Docker Swarm
Has a vast open-source community	Has a smaller community compared to Kubernetes
More extensive and customizable	Less extensive and customizable
Requires heavy setup	Easy to set up files
High fault tolerance	Low fault tolerance
Provides strong guarantees to cluster states, at the expense of speed	Facilitates for quick container deployment in large clusters
Manual Load balancing	Automatic Load balancing

FIGURE B.13 – Tableau Comparatif Kubernetes VS Docker Swarm

### B.3.6.3 Popularité Kubernetes VS Docker Swarm

En raison de son large support communautaire et de sa capacité à gérer même les scénarios de déploiement les plus complexes, Kubernetes est souvent le choix numéro un pour les équipes de développement d'entreprise qui gèrent des applications basées sur des microservices.

En ce qui concerne la popularité, Kubernetes a un net avantage, comme nous pouvons l'observer selon le graphique Google Trends. De plus, en regardant Github, nous pouvons conclure que si Kubernetes compte 81,1k étoiles, Docker Swarm n'en a que 5,8k. C'est une grande différence et ne laisse pas beaucoup de place au doute. Kubernetes est certainement une solution plus populaire que Docker Swarm en ce qui concerne les technologies d'orchestration de conteneurs. *Voir Graphe si dessous*

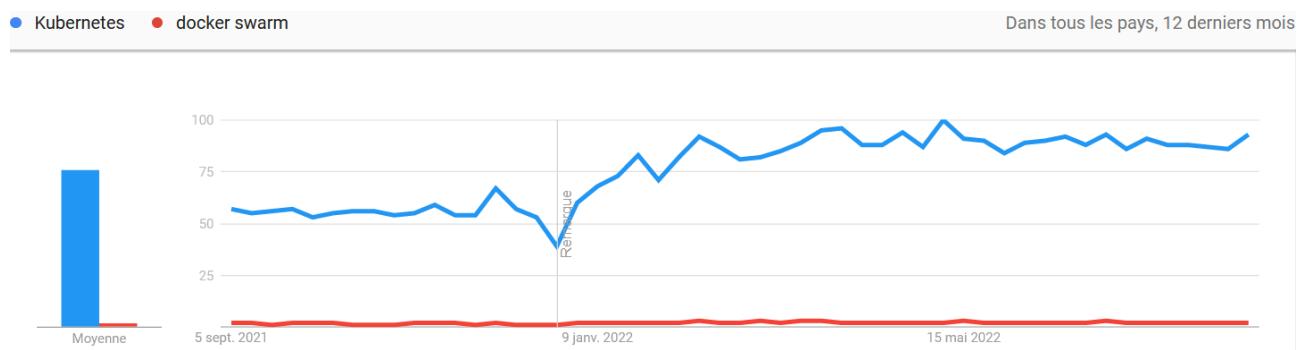


FIGURE B.14 – Popularité Kubernetes VS Docker Swarm -Google Trends-

#### **B.3.6.4 Récapitulatif De Comparaison**

Comme nous pouvons le voir, Docker Swarm et Kubernetes ont été créés pour remplir le même objectif.

1. Pour les débutants, Docker Swarm est une solution simple et facile à utiliser pour gérer les conteneurs à grande échelle. Si l'entreprise évolue vers le monde des conteneurs et n'a pas de charges de travail complexes à gérer, alors Docker Swarm est le bon choix.
2. Si les applications sont délicates et que l'entreprise recherche un package complet comprenant la surveillance, les fonctionnalités de sécurité, l'auto-réparation, la haute disponibilité et une flexibilité absolue, alors Kubernetes est le bon choix.

### B.3.7 Kubeadm

Selon la documentation de Microsoft, il existe trois manières de déployer un cluster Big Data :

1. Minikube
2. Kubeadm
3. AKS

Nous avons choisi Kubeadm. Il est utilisé pour amorcer les composants Kubernetes, et non pour provisionner les machines. Cette option est favorable si l'entreprise veut héberger le cluster Big Data sur site. Le déploiement se fait avec un minimum de 64 Go de RAM sur chaque hôte/nœud.

L'avantage de cette option est que l'entreprise peut contrôler totalement le cluster Kubernetes sous-jacent. Kubeadm est la « voie difficile » pour commencer avec Kubernetes car l'entreprise sera chargée non seulement des nœuds de travail, mais aussi du nœud maître. Donc, il faut renforcer les compétences en administration Kubernetes. La figure ci-dessous démontre la différence entre les 3 outils de déploiement de SQL SERVER BDC :

Options De Déploiement BDC	KUBEADM	MINIKUBE	AKS
<b>Installation</b>	Difficile	Très Facile	Facile
<b>Notion de Nodes</b>	Oui (Min 1 Node)	Cluster Minimale 1Node	Oui (Min 1 Node)
<b>Gestion De Nodes</b>	Manuel	/	Manuel
<b>HA availability</b>	Oui	Non	N/A
<b>Sécurité</b>	Manuel	Minimale	Supporté
<b>Capacité Machine</b>	Très Puissante	Moyenne	Moyenne
<b>Efficacité</b>	Oui	Non(Test)	Oui
<b>Extensible</b>	Oui	Non	Oui(Payant)
<b>Coût</b>	Gratuit	Gratuit	\$85.41 /Mois

FIGURE B.15 – Kubeadm VS MiniKube VS AKS

#### B.3.7.1 Pourquoi Kubeadm ?

- **Solution Gratuite** : Kubeadm est totalement gratuit.
- **Efficacité** Selon la documentation officielle, kubeadm peut être exploiter comme élément de base dans des systèmes complexes avec d'autres installateurs qui est notre cas .
- **Gestion De Cluster** : Nous avons la possibilité de gérer le cluster et de l'adapter selon les besoins.
- **Niveau de production de l'architecture complète** Permet de découvrir tout le potentiel de Kubernetes .

### B.3.8 Grafana

Grafana est une plate-forme open source pour surveiller, analyser et visualiser les données du système en temps réel. Le but de cette solution est d'afficher simplement et intuitivement de grandes quantités de données provenant de diverses sources.

Après réception des données, Grafana les analyse et les présente dans des tableaux de bord hautement personnalisables, intuitifs et faciles à lire.



FIGURE B.16 – Grafana Logo

#### B.3.8.1 Pourquoi Grafana ?

- **Solution Gratuite** : Grafana est Open-Source. Cela signifie que l'outil est gratuit et que nous pouvons l'enrichir avec notre propre code. Cela signifie également que nous avons une grande communauté de développeurs dédiés à l'amélioration du produit, à la correction des bogues et à la sécurité.
- **Multiplateformes et multisources de données** Grafana est disponible sur Linux, Mac, Windows. L'outil peut visualiser des données stockées dans plusieurs types de bases de données (Graphite, Prometheus, Influx DB, ElasticSearch, SQL SERVER, MySQL, PostgreSQL etc.)
- **Communauté** La puissance de cette solution est appréciée et approuvée par de nombreux développeurs, administrateurs système et entreprises. Des géants de la technologie comme Paypal, Intel et Stack Overflow font confiance à Grafana pour leurs systèmes d'analyse.
- **Adaptation Au Besoin** Grafana est conçu pour répondre exactement aux besoins de l'entreprise et au type de données qu'elle souhaite explorer. Nous pouvons créer et modifier des tableaux de bord à notre guise. Si cela ne suffit pas, Grafana offre la possibilité d'intégrer nos propres plugins pour encore plus de flexibilité.
- **Assure Différents Volets** : De plus Grafana permet de : observabilité unifiée, surveillance des conteneurs, un seul tableau de bord, plusieurs utilisateurs, résolution des problèmes opérationnels de manière collaborative, surveillance IoT, surveillance du cycle de vie du développement logiciel

### B.3.9 Git

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre et gratuit, créé en 2005. C'est le logiciel de contrôle de version le plus populaire pour le développement de logiciels et de sites Web, utilisé par des millions de personnes dans tous les environnements (Windows, Mac, Linux). Git est également le système de base du célèbre site Web GitHub, principal hébergeur de code informatique.

Git avec sa structure distribuée illustre parfaitement ce qu'est un système de contrôle de version distribué (DVCS). Au lieu de réservé un emplacement unique pour tout l'historique des versions logicielles, comme c'est souvent le cas dans les systèmes de contrôle de version traditionnels tels que CVS et Subversion (alias SVN), Git permet à chaque copie de travail du code de garder une trace de toutes les modifications. L'histoire complète de Outre la décentralisation, Git a été conçu pour atteindre trois objectifs : performances, sécurité et flexibilité.



FIGURE B.17 – Git Logo

### B.3.10 Gogs

Gogs est une interface Web pour le Git Release Manager. Il est très léger et fonctionne très bien avec les micro-ordinateurs de type Raspberry Pi. Très utile pour l'auto-hébergement de divers projets. Gogs a une interface similaire à GitHub.



FIGURE B.18 – Gogs Logo

#### B.3.10.1 Pourquoi GOGS ?

- **Solution Économique** : Gogs est open source. Par conséquent, il n'y a pas de frais de licence à l'achat.
- **Compatibilité** : Gogs est multiplateforme : Linux , Windows , MacOS
- **Gestion d'utilisateur** : Organisation , équipes et collaborateurs
- **Gestion De Projet** : Issue Tracker , milestone , version etc
- **Léger** : Gogs a de faibles exigences minimales et peut fonctionner sur un Raspberry Pi peu coûteux. Certains utilisateurs exécutent même des instances Gogs sur leurs appareils NAS.

### B.3.11 Windows Server 2019 -Datacenter-

Les serveurs sont des machines dont le fonctionnement et les cas d'usage diffèrent des ordinateurs personnels. C'est la raison pour laquelle ils nécessitent des systèmes d'exploitation adaptés. Microsoft Windows Server est le système d'exploitation serveur de Microsoft. En quelque sorte, il s'agit d'une version améliorée du système Windows standard .

Sa tarification se base elle aussi sur le nombre de coeurs des serveurs physiques (aux environs de 6 500 €), et nécessite également une licence d'accès client pour chaque machine cliente (toujours en supplément d'une licence Windows Pro).



FIGURE B.19 – Windows Server 2019 -Datacenter- Logo

#### B.3.11.1 Pourquoi Windows Server 2019 -Datacenter- ?

- **Haute Gamme** Avec cette édition , nous avons plus de fonctionnalités que les autre éditions Windows Server .
- **Hyper-V** : Il est possible de créer autant de machines virtuelles que nécessaire sous Windows Server. Elle se destine aux entités ayant de forts besoins de virtualisation, pour la mise en œuvre de centres de données totalement gérés logiciellement
- **Conteneurisation**  
Windows Server 2019 Standard peut exécuter un nombre illimité de conteneurs Windows, mais seulement deux conteneurs Hyper-V. Windows Server 2019 Datacenter permet d'exécuter des conteneurs Windows illimités et des conteneurs Hyper-V illimités.
- **services Orientés Serveur** : En tant que tel, il offre une variété de services orientés serveur, tels que la capacité d'héberger des sites Web, de gérer les ressources entre différents utilisateurs et applications, ainsi que des fonctions de messagerie et de sécurité. Compatible avec la plupart des langages de programmation Web et des systèmes de bases de données, y compris .NET Core, ASP.NET, PHP, MySQL et MS SQL.

### B.3.12 Hyper-V

Hyper-V, également connu sous le nom de Windows Server Virtualization, est un système de virtualisation. Il transforme un serveur physique en hyperviseur, lui permettant de gérer et d'héberger des machines virtuelles, communément appelées VM (machines virtuelles). Hyper-V permet aux utilisateurs de Windows de démarrer leurs propres machines virtuelles. Cette machine virtuelle permet de virtualiser l'infrastructure matérielle complète, y compris la RAM, l'espace disque, la puissance du processeur et d'autres composants. Un autre système d'exploitation fonctionne par-dessus, mais il n'est pas nécessaire que ce soit Windows. Par exemple, il est très courant d'exécuter des distributions open source de Linux sur des machines virtuelles.



FIGURE B.20 – Hyper-V Logo

#### B.3.12.1 Pourquoi Hyper-V

- **Une Meilleure Disponibilité :** Toutes les solutions de virtualisation d'aujourd'hui permettent la migration à chaud des machines virtuelles. Avec la possibilité de déplacer des machines virtuelles d'un serveur physique à un autre sans arrêter les machines virtuelles. Cette fonctionnalité est un facteur important dans l'amélioration de la disponibilité du service. L'utilisation de deux serveurs physiques permet ainsi une simple duplication (redondance) de l'infrastructure de virtualisation. Si l'un des deux serveurs tombe en panne, les VM sont automatiquement déplacées vers le deuxième serveur.
- **De Meilleures Performances :** La migration en direct des machines virtuelles entre les serveurs physiques présente un autre avantage. Cela signifie que la charge de travail peut être répartie entre les serveurs. Si la charge d'une machine virtuelle augmente de manière significative, d'autres machines virtuelles peuvent se rabattre sur un serveur physique moins utilisé.
- **Autonomie :** Les machines virtuelles sont également autonomes. Ainsi, par exemple, l'exécution d'un logiciel qui plante le système ne compromet pas le périphérique physique. Seule la machine virtuelle doit être réinitialisée.
- **Avantage Pour Les Développeurs De Logiciels :** Les programmes que crées peuvent être testés dans diverses conditions logicielles et matérielles. De plus, avec une machine virtuelle autonome, les développeurs ne se soucient pas des codes bogués qui endommage le système.

## B.4 Conclusion

Dans cette partie nous avons justifié notre choix technologique et sa valeur ajoutée dans la CASNOS. En ce qui suit ,nous allons présenter les étapes de déploiement de la solution.