

Rapport de recommandation

Migration de l'hébergement de l'application GEOD'AIR



L'Innovation Scientifique & Technique



A l'attention de :

Jonathan BARRIER, INERIS

Laurent LETINOIS, INERIS

Laure MALHERBE, INERIS

Rédigé par :

Benjamin LECLERC, IT link

Maxime LECLERC, IT link

Approuvé par :

Emmanuel Camus, Directeur de projet, IT link.

Préambule

Ce document constitue l'offre technique et commerciale pour l'étude de faisabilité dans le cadre de la migration de l'hébergement.

Sommaire

1. Analyse de l'existant.....	4
1.1. Architecture et flux de données	4
1.2. Dimensionnement des VM.....	6
1.3. Composants logiciels et versions.....	11
1.4. Versions des middlewares.....	12
2. Propositions d'ajustements et recommandations	15
2.1. Conseils généraux d'architecture	15
2.2. Scénario d'évolution 1.....	15
2.3. Scénario d'évolution 2.....	18
2.4. Scénario d'évolution 3.....	21
2.5. Scénario d'évolution 4.....	23
2.6. Alternative aux scénarios.....	25
2.7. Bilan des solutions proposées	26
3. Organisation de l'exploitation	29
3.1. Rôles de SPIE	29
3.2. Rôles d'IT Link	29
4. Ébauche du plan de migration	31

1. Analyse de l'existant

La plateforme GEOD'AIR est aujourd'hui déployée sur les serveurs de l'hébergeur AGARIK. Le contrat d'hébergement prenant fin courant Mai 2018, l'INERIS souhaite migrer l'application chez l'hébergeur SPIE. Pour effectuer la migration dans les meilleures conditions possibles, et assurer la pérennité de GEOD'AIR, l'INERIS souhaite aussi analyser les scénarios d'évolutions de l'architecture, pour l'optimiser lors de cette transition.

1.1. Architecture et flux de données

Ci-dessous un diagramme représentant les différents flux entre les services. Une description exhaustive des flux est réalisée dans le DAT futur basé sur la fourniture de l'Excel de l'INERIS sur ce même sujet.

L'accès Internet et par ssh aux VMs se fait à travers Web1, qui sert de « proxy » aux autres VMs. Les VMs ont ensuite un accès ssh autorisé entre elles. Ce système d'accès à travers Web1 sera reconduit à priori suite aux discussions avec SPIE.

La VM DAT1 a en plus des connexions FTP venant de serveurs client, tel que l'UE ou les ASQAA, et une connexion HTTP vers data.gouv.fr à travers un tunnel ssh.

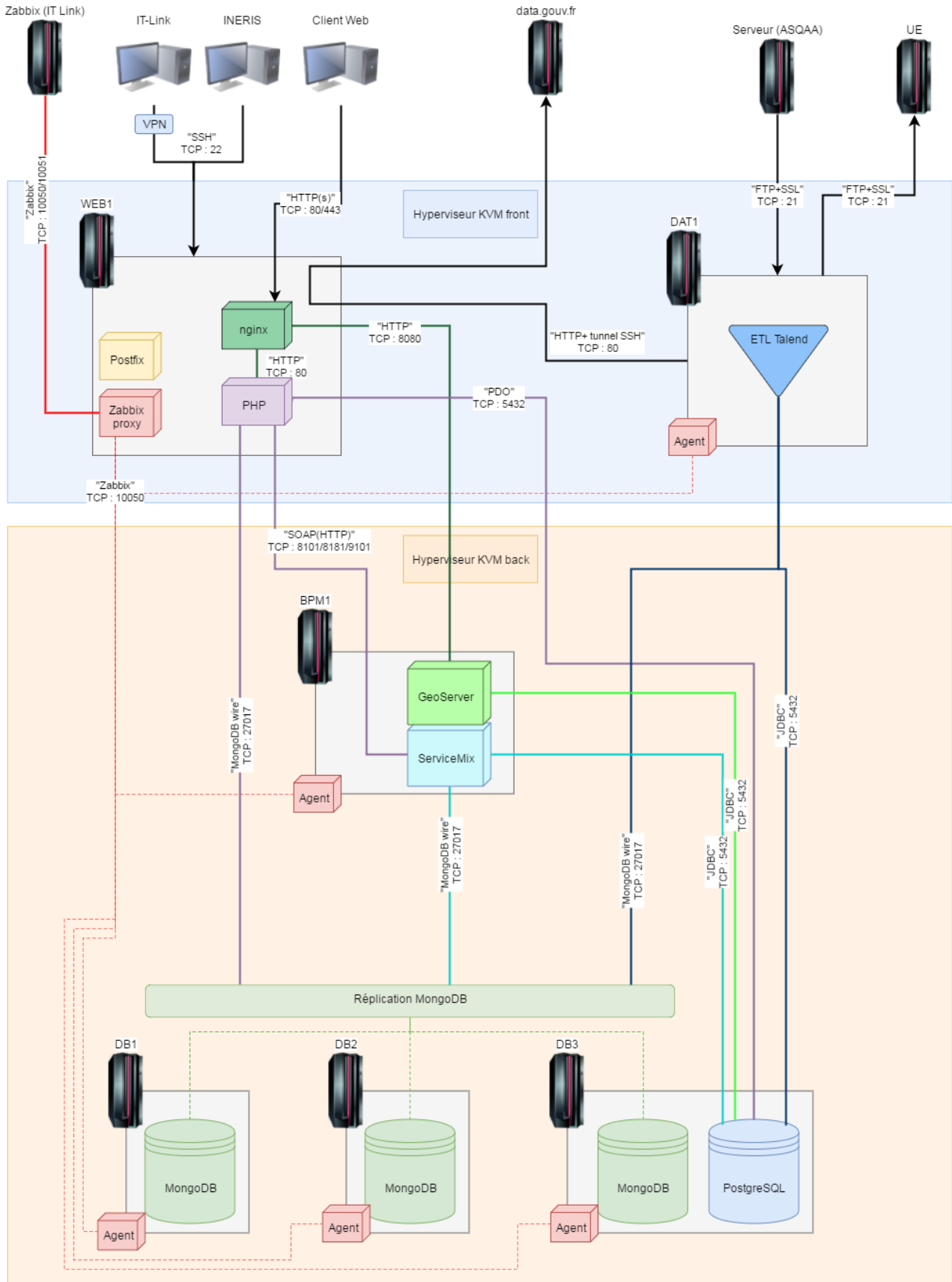


Figure 1 : Flux de données entre les services existants

Les flux mail ne sont pas représentés dans la figure ci-dessus mais seront dans le dossier d'architecture technique.

1.2. Dimensionnement des VM

La partie suivante présente l'état actuel des dimensions des VMs utilisées par GEOD'AIR.

1.2.1. Hardware actuel

Les CPU sont virtuels et mappés a priori sur un CPU physique. La RAM disponible correspond à la RAM visible par chaque VM (à l'aide de Zabbix ou de la commande `free`). L'espace disque est estimé à partir de la somme de chaque volume ajouté à la VM par l'hyperviseur, et correspond aux répertoires disponibles pour chaque VM.

Tableau 1 : Dimensionnement de chaque VM.

VM	CPU (#cœurs)	RAM (Go)	Disque HDD (Go)
WEB1	4	16	15
DAT1	20	64	216
BPM1	6	16	55
DB1	8	32	267
DB2	8	32	316
DB3	10	32	296
Total	56	192	1165

1.2.2. Utilisation des VM

En moyenne, l'infrastructure n'est pas beaucoup sollicitée (Tableau 2). De fait, la majorité des services passe une majorité du temps en phase d'attente.

Tableau 2 : Moyenne d'utilisation matériel sur 1 mois

VM	CPU (%)	RAM (Go)	Disque HDD (Go)
WEB1	1%	10	6
DAT1	10%	6	166
BPM1	0.5%	8	18
DB1	2%	8	246
DB2	1%	16	254

DB3	0.5%	21	210
Total	N/A	69	900

Le pic de charge le plus fort observé à ce jour correspond notamment à la réintégration des données annuelles effectué en novembre. Cette opération est l'opération la plus coûteuse effectué avec GEOD'AIR. On peut donc référencer ce pic pour l'utilisation matériel maximale de l'application (Tableau 3).

Tableau 3 : Utilisation matériel maximale enregistrée sur 1 mois

VM	CPU (%)	RAM (Go)	Disque HDD (Go)
WEB1	10%	14	6
DAT1	80%	22	166
BPM1	15%	9	18
DB1	26%	25	246
DB2	3%	28	254
DB3	20%	29	255
Total	N/A	127	945

D'un point de vue volumétrie, l'utilisation réseau est résumée dans les tableaux suivants (en Kbps) :

Tableau 4 : Entrées/sorties réseau pour chaque VM en moyenne sur 3 mois.

VM	Trafic entrant (Kbps)	Trafic sortant (Kbps)
WEB1	271	257
DAT1	2 852	1 320
BPM1	76	65
DB1	398	2 493
DB2	205	486
DB3	266	551

Total	4068	5172
-------	------	------

Utilisation pic sur 3 mois (métrique calculée en sommant le trafic pic de chaque interface réseau, or ces pics sont rarement simultanés, le tableau ci-dessous est donc le trafic théorique maximal résultant de pics simultanés.) :

Tableau 5 : Entrées/sorties réseau pour chaque VM, pic de charge le plus fort sur 3 mois.

VM	Trafic entrant (Kbps)	Trafic sortant (Kbps)
WEB1	190 650	186 020
DAT1	182 430	108 780
BPM1	44 960	120 880
DB1	535 720	94 190
DB2	253 380	624 290
DB3	479 060	626 470
Total	1 686 200	1 760 630

1.2.3. Agrégation des données de CPU

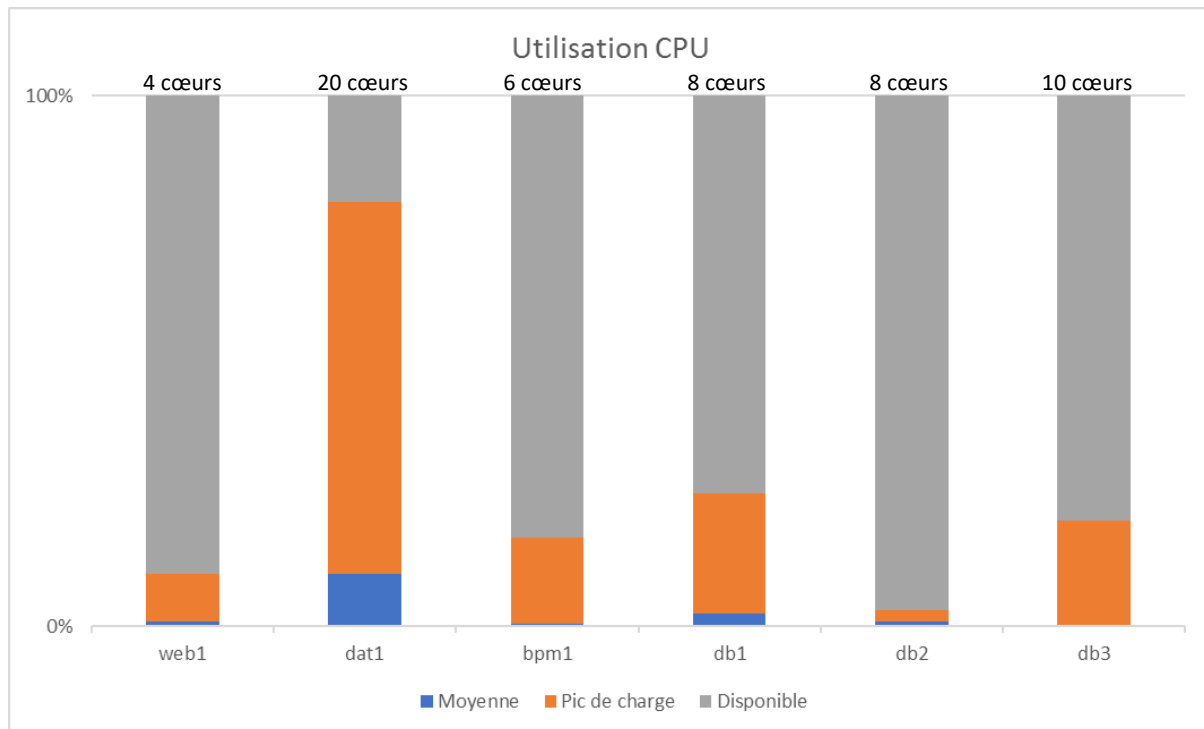


Figure 2 : Utilisation moyenne (bleu) et maximale (orange) du CPU pour chaque VM en relation à la capacité maximale de la VM.

Pour avoir un dimensionnement idéal, on cherche à avoir une utilisation CPU à environ 80% en pic de charge pour permettre aux application d'avoir de la marge et d'assurer la disponibilité de la VM lors d'une forte utilisation.

Sur ce graphe (Figure 2), on peut voir que le dimensionnement des CPU est largement surdimensionné, même en pic de charge, à l'exception de la VM DAT1.

1.2.4. Agrégation des données de RAM

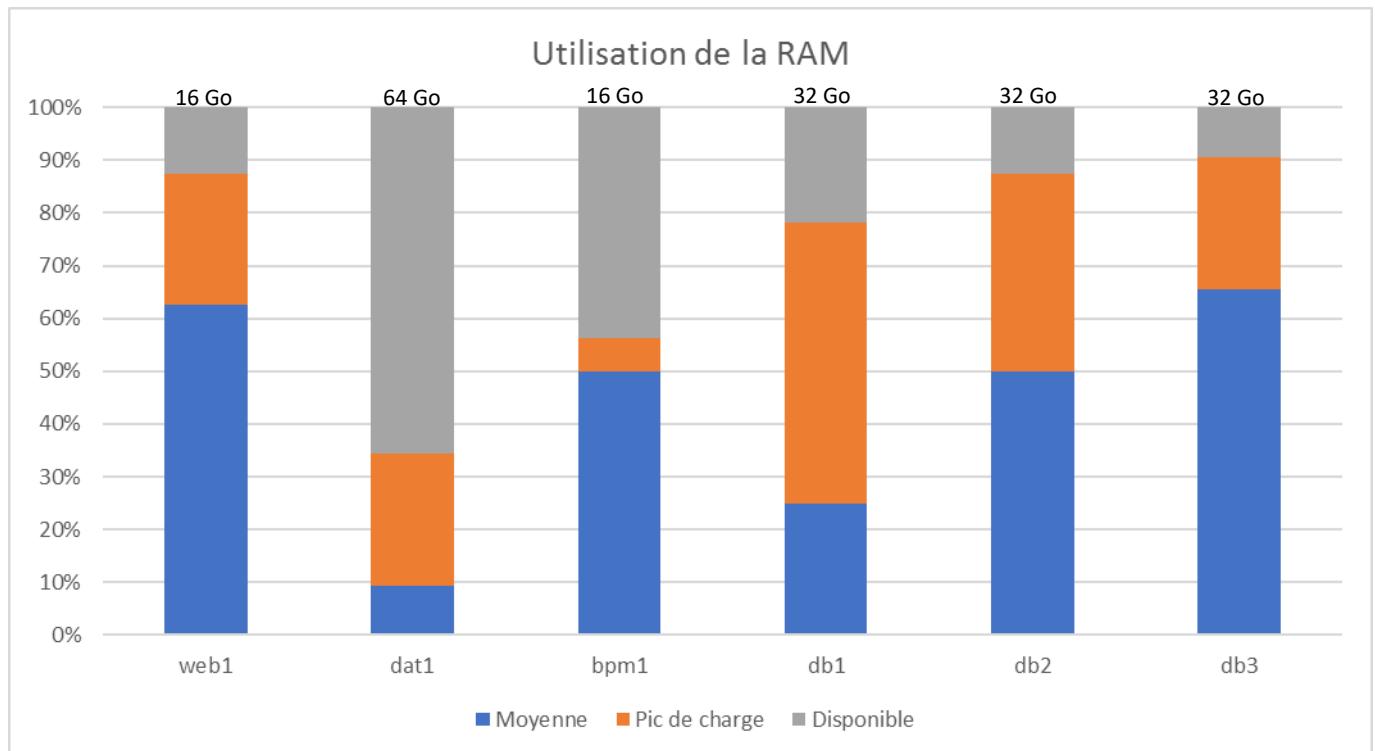


Figure 3 : Utilisation moyenne (bleu) et maximale (orange) de la RAM pour chaque VM en relation à la capacité maximale de la VM.

Pour avoir un dimensionnement idéal, on cherche à avoir une utilisation de la RAM maximale en pic de charge (environ 90%) avec une faible marge pour être certain que l'application utilise un maximum de RAM, sans être limitée par celle-ci.

Sur ce graphe (Figure 3), on peut voir que le dimensionnement de la RAM est surdimensionné pour DAT1 et BPM1, et relativement correct pour les autres VM.

1.2.5. Agrégation des données de stockage

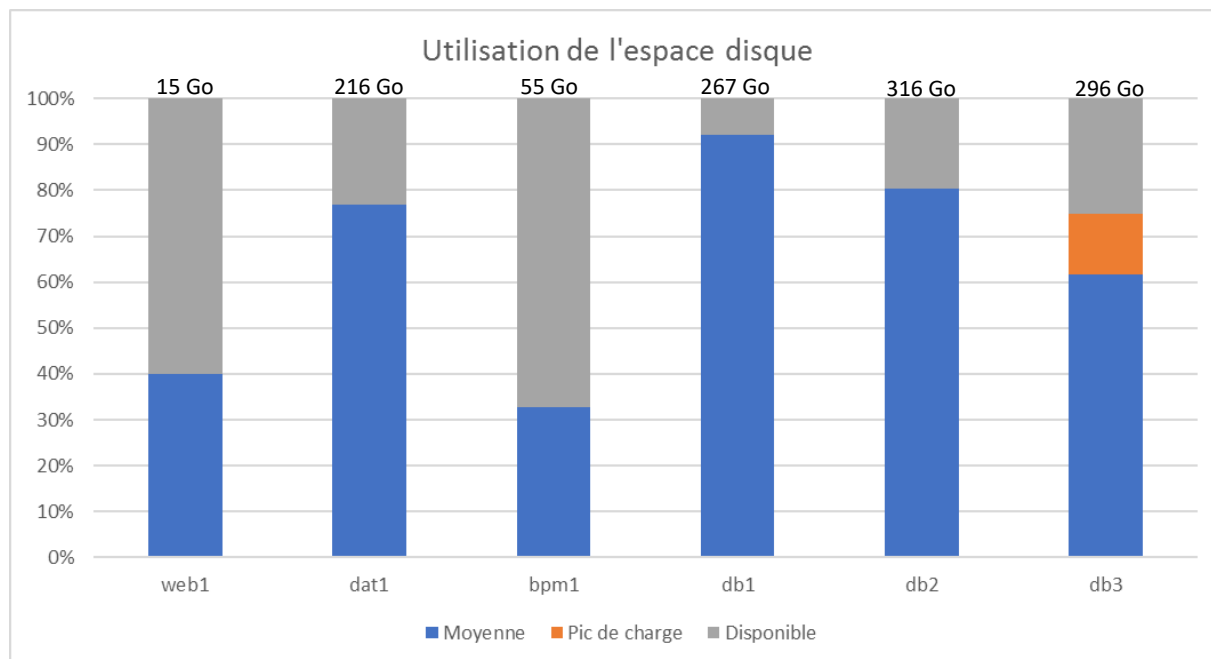


Figure 4 : Utilisation moyenne (bleu) et maximale (orange) de l'espace disque pour chaque VM en relation à la capacité maximale de la VM.

Pour avoir un dimensionnement idéal, on cherche à avoir une utilisation de l'espace disque relativement moyen (environ 50%), pour avoir suffisamment de place pour d'éventuelles opérations (création d'archives, ajouts de nouvelles données, etc.)

Sur ce graphique (Figure 4), on peut voir que le dimensionnement de l'espace disque est correcte sur WEB1 et BPM1. L'espace disque commence à poser un risque sur DAT1. L'espace disque est insuffisant sur les VMs DB1/2/3.

1.3. Composants logiciels et versions

La partie suivante présente les versions actuelles des middlewares pour GEOD'AIR V2.1.X et V3.0.X, ainsi que les dernières versions ou LTS vers lesquelles on peut évoluer.

Tableau 6 : Composants logiciels et versions correspondantes.

Machine virtuelle (VM)	OS GEOD'AIR	Middleware	GEOD'AIR V2.1.X	GEOD'AIR V3.0.X	Dernier version ou LTS
WEB1	CentOS 6.4 64 bits	NGINX	1.4.5	1.4.5	1.15.7
WEB1	CentOS 6.4 64 bits	PHP	5.3.3	5.6	7.1+
WEB1	CentOS 6.4 64 bits	POSTFIX	2.6.6	2.6.6	3.3.2
BPM1	CentOS 6.4 64 bits	JAVA	1.7.0_17	1.7.0_17	11

BPM1	CentOS 6.4 64 bits	Servicemix	4.5.3	4.5.3	7.0.1
BPM1	CentOS 6.4 64 bits	GEOSERVER	2.4.3	2.4.3	2.13.3
BPM1	CentOS 6.4 64 bits	Bonita	6.2.2	6.2.2	7.5.x
DB1	CentOS 6.4 64 bits	MongoDB	2.4.9	2.4.9	4.x
DB2	CentOS 6.4 64 bits	MongoDB	2.4.9	2.4.9	4.x
DB3	CentOS 6.4 64 bits	MongoDB	2.4.9	2.4.9	4.x
DB3	CentOS 6.4 64 bits	PostgreSQL avec Postgis	Postgresql 9.3.2 Postgis 2.1.1 r12113	Postgresql 9.3.2 Postgis 2.1.1 r12113	10
DAT1	CentOS 6.4 64 bits	VSFTPD	2.2.2	2.2.2	3.0.3
DAT1	CentOS 6.4 64 bits	JAVA	1.7.0_17	1.7.0_17	11

1.4. Versions des middlewares

Dans cette partie nous étudions le coût et les subtilités liées aux possibles montés en version plus récente ou LTS.

1.4.1. NGINX (1.4 -> 1.15)

- Changelog : Nombreux patches de features et de sécurité
- Complexité : 1/10
- Points d'attention : La syntaxe du fichier de configuration évolue régulièrement, il faudra probablement le réécrire.

1.4.2. PHP (5.6 -> 7.1)

- Changelog : Nouveau moteur (PHP#NG, 2x plus performant), nombreux features pour faciliter le code, nettoyage des APIs dépréciées
- Complexité : 10/10
- Points d'attention : Le nettoyage des APIs a permis une remise au propre du core de PHP au détriment des projets utilisant les vieilles API dépréciées. Entre les quelques changements de syntaxe, et les nombreux changements d'API, porter l'application en PHP 7+ implique un recodage considérable de la base de code.

1.4.3. POSTFIX (2.6 -> 3.3)

- Le service Postfix est pris en charge par un service mutualisé SPIE.

1.4.4. JAVA (1.7 -> 11)

- Changelog : 11 est la prochaine version LTS (le 8 se terminant le 01/2019), nouveaux features de performance et de sécurité
- Complexité : 6/10
- Points d'attention : Lors d'une mise à jour récente, les APIs JEE ont été enlevées, et doivent être explicitement importées par Maven. Aussi, certains frameworks devraient être mis à jour vers des versions récentes pour rester compatible, tel que spring et hibernate. Les versions plus récentes de ces frameworks risquent par contre d'avoir un impact sur la connectivité des modules (différentes APIs, etc.). Certains modules doivent d'ailleurs rester en Java 8, comme Talend ou GeoServer par exemple.
- Concerne Talend (ETL) et le module statistique.

1.4.5. SERVICEMIX (4.5 -> 7.0)

- Changelog : Dépendances de Karaf sont lourdement modifiées.
- Complexité : 7/10
- Points d'attention : Non compatible Java 11, Java sera donc monté qu'en Java 8 au maximum.

1.4.6. GEOSERVER (2.4 -> 2.13)

- Changelog : Bugfix et améliorations
- Complexité : 1/10
- Points d'attention : Backup les données avant d'effectuer la migration, certaines versions peuvent avoir des formats incompatibles.

1.4.7. BONITA

- À supprimer. En effet, ce module existe pour des raisons historiques mais n'a plus d'utilité.

1.4.8. MONGODB (2.4 -> 4.0)

- Changelog : Nouveau moteur (WiredTiger), optimisations, changement de certaines APIs
- Complexité : 5/10
- Points d'attention : Certaines commandes ne sont plus supportées, certains fichiers de configuration ont des options différentes. Les connecteurs à Java et PHP ne sont plus les mêmes, ce qui implique une révision des dépendances et une mise à jour du code.

1.4.9. POSTGRESQL (9.3 -> 9.6)

- Changelog : Nombreuses optimisations de performance
- Complexité : 1/10
- Points d'attention : Quelques fonctions ne sont plus supportées

1.4.10. VSFTPD (2.2 -> 3.0)

- Changelog : Bugfix et patch de certains comportements de sessions SSL
- Complexité : 1/10
- Points d'attention : Des timeouts de session SSL ont changés

1.4.11. Tableau de synthèse

Tableau 7 : Synthèse des versions de logiciel

Middleware	GEOD'AIR V3.0.X	Dernier version ou LTS	Complexité
NGINX	1.4.5	1.15.7	1/10
PHP	5.6	7.1+	10/10
POSTFIX	2.6.6	3.3.2	N/A
JAVA	1.7.0_17	11-13	8/10
		8	6/10
Servicemix	4.5.3	7.0.1	7/10
GEOSERVER	2.4.3	2.13.3	1/10
Bonita	6.2.2	7.5.x	N/A
MongoDB	2.4.9	4.x	5/10
PostgreSQL avec Postgis	Postgresql 9.3.2 Postgis 2.1.1 r12113	10	2/10
VSFTPD	2.2.2	3.0.3	1/10

2. Propositions d'ajustements et recommandations

2.1. Conseils généraux d'architecture

2.1.1. Briques services

D'un point de vue haut niveau, l'architecture pose des problèmes d'isolation des services. En effet, autant pour des raisons de sécurité que de performance, il est préférable d'isoler au maximum chaque service.

Dans l'architecture existante, on peut voir que plusieurs services partagent les mêmes environnements et ressources :

- Nginx, PHP, Postfix et Zabbix
- GeoServer et ServiceMix
- MongoDB et PostgreSQL

De plus, le seul service hautement disponible est actuellement mongoDB. Pour une architecture de production, il est préférable d'avoir un système de « failover » pour les services critiques.

Dans l'architecture existante, si par exemple l'ETL ne fonctionne plus, le service est indisponible jusqu'à une intervention manuelle. De même pour le reste des briques services en dehors de mongoDB.

Il faut donc bien identifier le besoin de disponibilité de chaque service et mettre en place une solution appropriée.

2.2. Scénario d'évolution 1

Le premier scénario d'évolution propose l'architecture la plus similaire à l'existant, offrant des performances et un niveau de service quasi similaire.

2.2.1. Ajustement de l'architecture

Pour l'architecture l'on retrouve les VMs existantes, avec quelques nouvelles VMs pour une meilleure séparation des services et des ressources. Notamment, PostgreSQL ne partage plus DB3 avec mongoDB, et Zabbix dispose d'une VM dédiée qui permet d'assurer le monitoring même si WEB1 tombe comme on peut le voir en Figure 5. Le service Postfix est remplacé par le service mutualisé de SPIE.

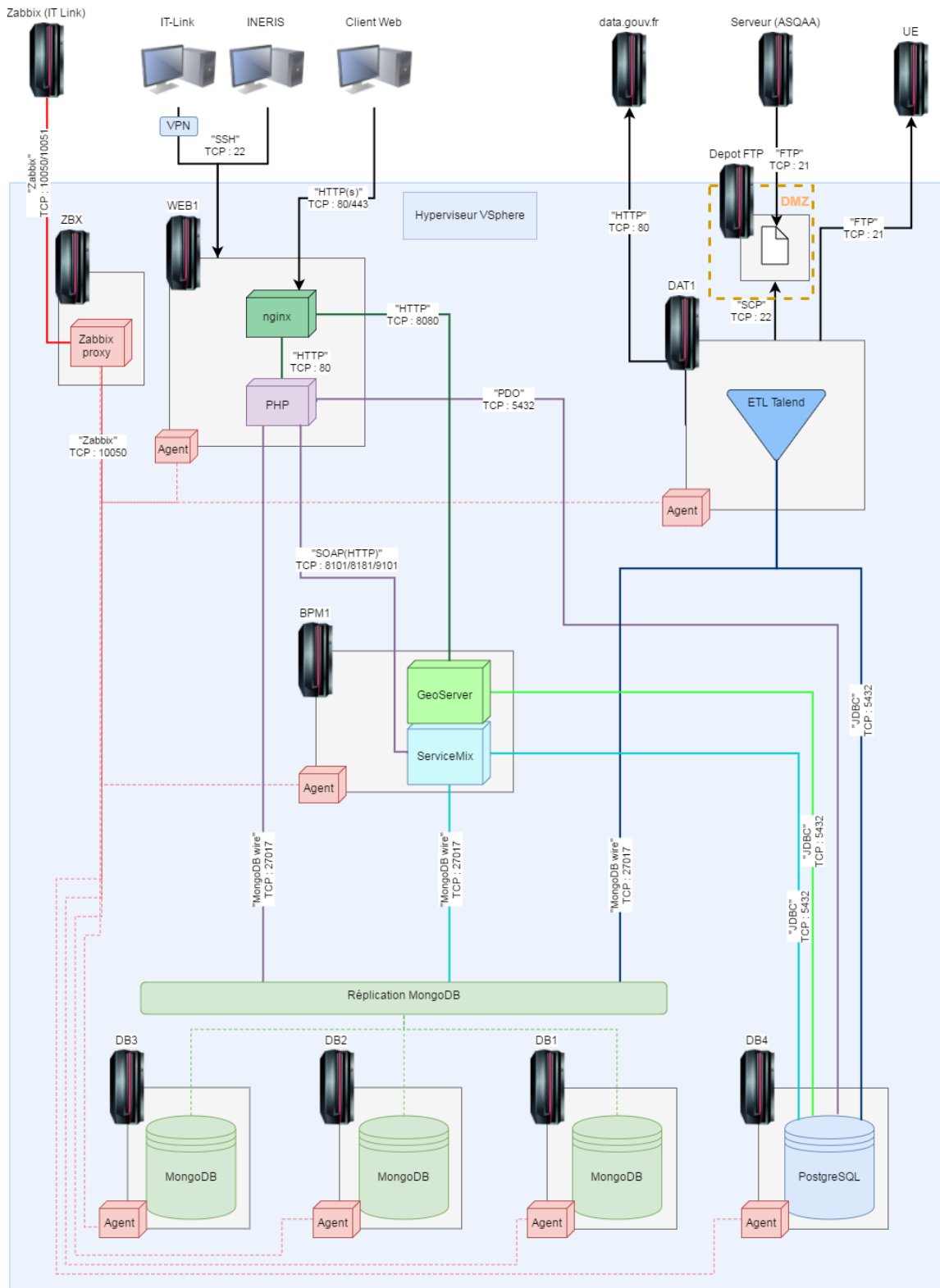


Figure 5 : Diagramme de flux pour le Scénario 1, on remarque notamment la séparation des VMs pour PostgreSQL et Zabbix.

Les flux mail ne sont pas représentés dans la figure ci-dessus mais seront dans le dossier d'architecture technique.

2.2.2. Dimensionnement des VM avec une Recette et une Production ISO

Un dimensionnement similaire est conservé, en ajustant selon l'utilisation actuelle. Les évolutions futures sont aussi considérées. En effet, les rapports ne seront plus conservés directement sur disque sur BPM1 mais intégrés sous forme de blob dans PostgreSQL. De plus l'évolution concernant le projet de pesticides augmente considérablement la taille des données mongoDB.

Tableau 8 : Dimensionnement de chaque VM.

VM	CPU (#cœurs)	RAM (Go)	Disque HDD (Go)
WEB1	4	16	20
ZBX	1	2	10
DAT1	20	32	200
BPM1	2	8	30
DB1	4	32	600
DB2	4	32	600
DB3	4	32	600
DB4	2	4	100
Total	41	158	2160

2.2.3. Dimensionnement des VM avec une Recette et une Production non ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2.

Tableau 9 : Dimensionnement des VMs

VM	CPU (#cœurs)	RAM (Go)	Disque Production (Go)	Disque Recette (Go)
WEB1	4	16	20	20
ZBX	1	2	10	10
DAT1	20	32	200	200
BPM1	2	8	30	30
DB1	4	32	600	200
DB2	4	32	600	200
DB3	4	32	600	200

DB4	2	4	100	100
Total	41	158	2160	960

2.2.4. Composant logiciels et versions attendus

L'INERIS, SPIE et IT Link se sont accordés pour monter en version les composants suivants (voir Tableau 7 : Synthèse des versions de logiciel) :

- CentOS 6.4 -> 7.4
- PostgreSQL

Pour les autres composants, les montées en versions sont faites uniquement pour ceux ayant une complexité simple n'entraînant pas de travaux majeurs sur l'application :

- Nginx
- GeoServer

Les montées en versions des composants restant seront effectuées par la suite au cours de phases successives.

2.3. Scénario d'évolution 2

Dans un second temps, une architecture conteneurisée utilisant Docker est envisageable. Cette architecture offre des performances et un niveau de service similaire au Scénario d'évolution 1.

2.3.1. Ajustement de l'architecture

Le but de cette architecture est de profiter de la technologie des conteneurs. Les services de l'ancienne VM BPM1 sont regroupés sur WEB1, et restent séparés grâce à l'isolation des conteneurs. En revanche cette isolation est purement logicielle. C'est pourquoi dans ce scénario et les prochains, l'ETL est toujours sur une VM séparée de la partie web. Dans le cas d'évènements (panne, maintenance...) sur une partie, l'autre ne doit pas être affectée.

Une des raisons principales d'introduire des conteneurs dans l'infrastructure est pour la maintenance. En effet, lorsque la configuration est terminée, il est plus simple de redéployer des conteneurs que de redéployer directement des applications. Notamment, le conteneur permet d'assurer que l'environnement autour d'un service est identique à toutes les étapes (si l'application fonctionne en développement, elle devrait fonctionner en production). En plus, l'environnement proposé par le conteneur est facilement évolutif en ce qui concerne les montées en version des middlewares.

En plus, les conteneurs permettent une gestion granulaire des services. Il est possible d'utiliser un orchestrateur tel que Kubernetes ou Docker Swarm pour gérer l'automatisation du redémarrage des services, la distribution de charge, le partitionnement des ressources, et la scalabilité. Pour le moment aucun orchestrateur n'est prévu, mais il peut/sera intégré par la suite.

Finalement, les bases de données restent sur des VM séparées pour assurer la plus haute disponibilité de la base possible.

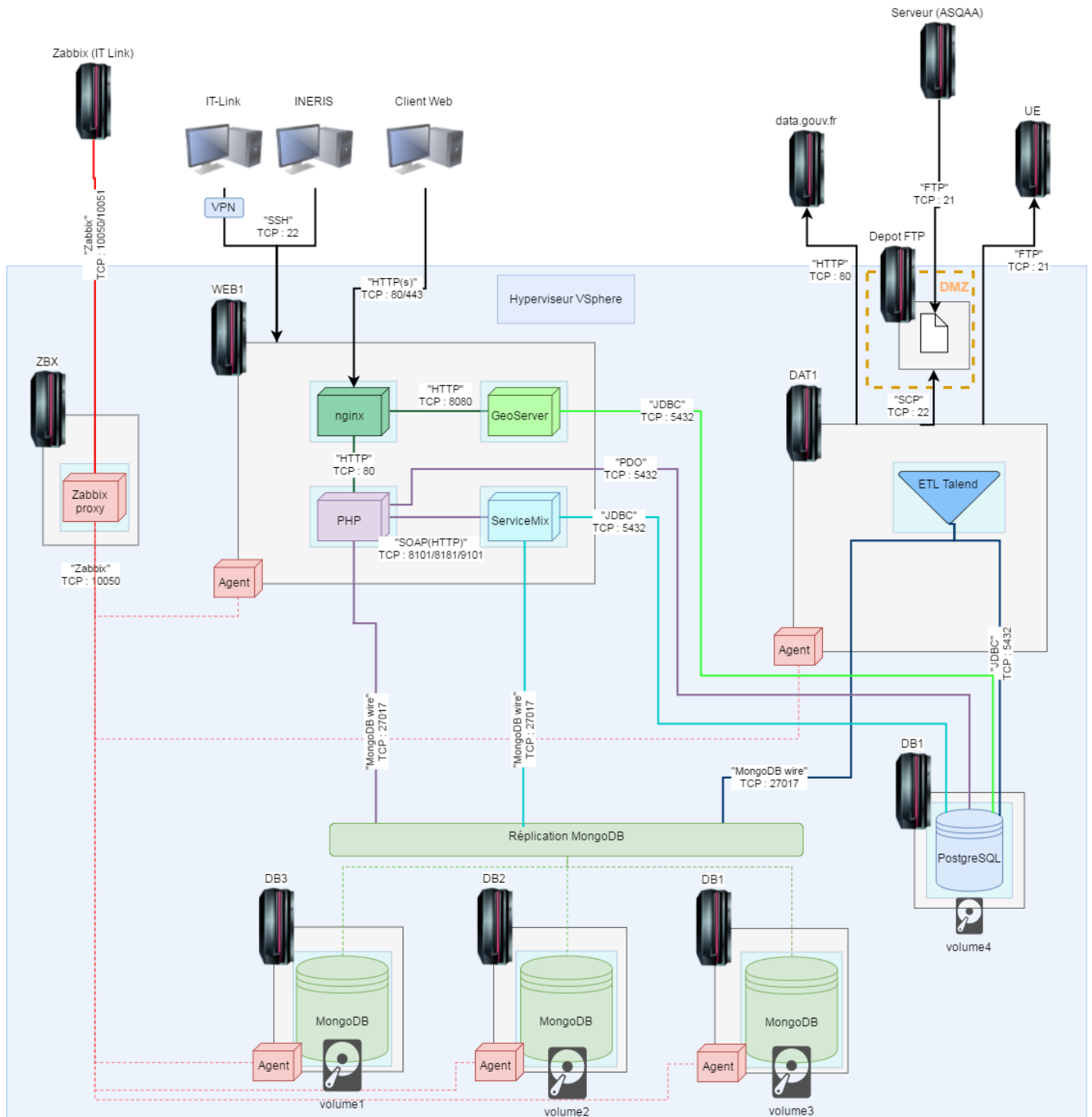


Figure 6 : Scénario 2, les middlewares sont conteneurisés. Les volumes représentent des espaces disques persistants alloués au conteneur.

Les flux mail ne sont pas représentés dans la figure ci-dessus mais seront dans le dossier d'architecture technique.

2.3.2. Dimensionnement des VM avec une Recette et une Production ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2.

Tableau 10 : Dimensionnement de chaque VM.

VM	CPU (#cœurs)	RAM (Go)	Disque HDD (Go)
WEB1	6	24	50
DAT1	20	32	200
ZBX	1	2	10
DB1	4	32	600
DB2	4	32	600
DB3	4	32	600
DB4	2	4	100
Total	41	158	2160

2.3.3. Dimensionnement des VM avec une Recette et une Production non ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2.

Tableau 11 : Dimensionnement des VMs

VM	CPU (#cœurs)	RAM (Go)	Disque Production (Go)	Disque Recette (Go)
WEB1	6	24	50	50
DAT1	20	32	200	200
ZBX	1	2	10	10
DB1	4	32	600	200
DB2	4	32	600	200
DB3	4	32	600	200
DB4	2	4	100	100
Total	41	158	2160	960

2.3.4. Composant logiciels et versions attendus

La solution conteneurisée facilite notamment les montées en version des logiciels. En effet, si l'on souhaite modifier une version de middleware, il suffit de modifier la configuration du conteneur. Le travail d'intégration/mise à niveau du code reste le même cependant. Lorsque l'application fonctionne, une image docker est créée et déployée en production.

Les versions attendues sont les mêmes qu'en 2.2.4.

2.4. Scénario d'évolution 3

Finalement, une troisième solution est de réduire au maximum le nombre de VM utilisé. Cette architecture conserve des performances similaires aux autres scénarios mais offre un niveau de service considérablement plus faible.

2.4.1. Ajustement de l'architecture

Le but de cette architecture est de réduire le nombre de VM en échange de disponibilité des services, notamment des bases de données.

Les bases peuvent-être regroupées sur une même VM plus puissante. En allouant des volumes persistants différents, il est possible de segmenter un disque entre les conteneurs, mais le mieux reste de séparer les volumes sur des disques différents, dans le but de conserver des performances de lecture/écriture élevées.

À noter que la documentation officielle de mongoDB déconseille fortement le déploiement de nœuds sur la même VM. (« <https://docs.mongodb.com/manual/tutorial/deploy-replica-set/> »)

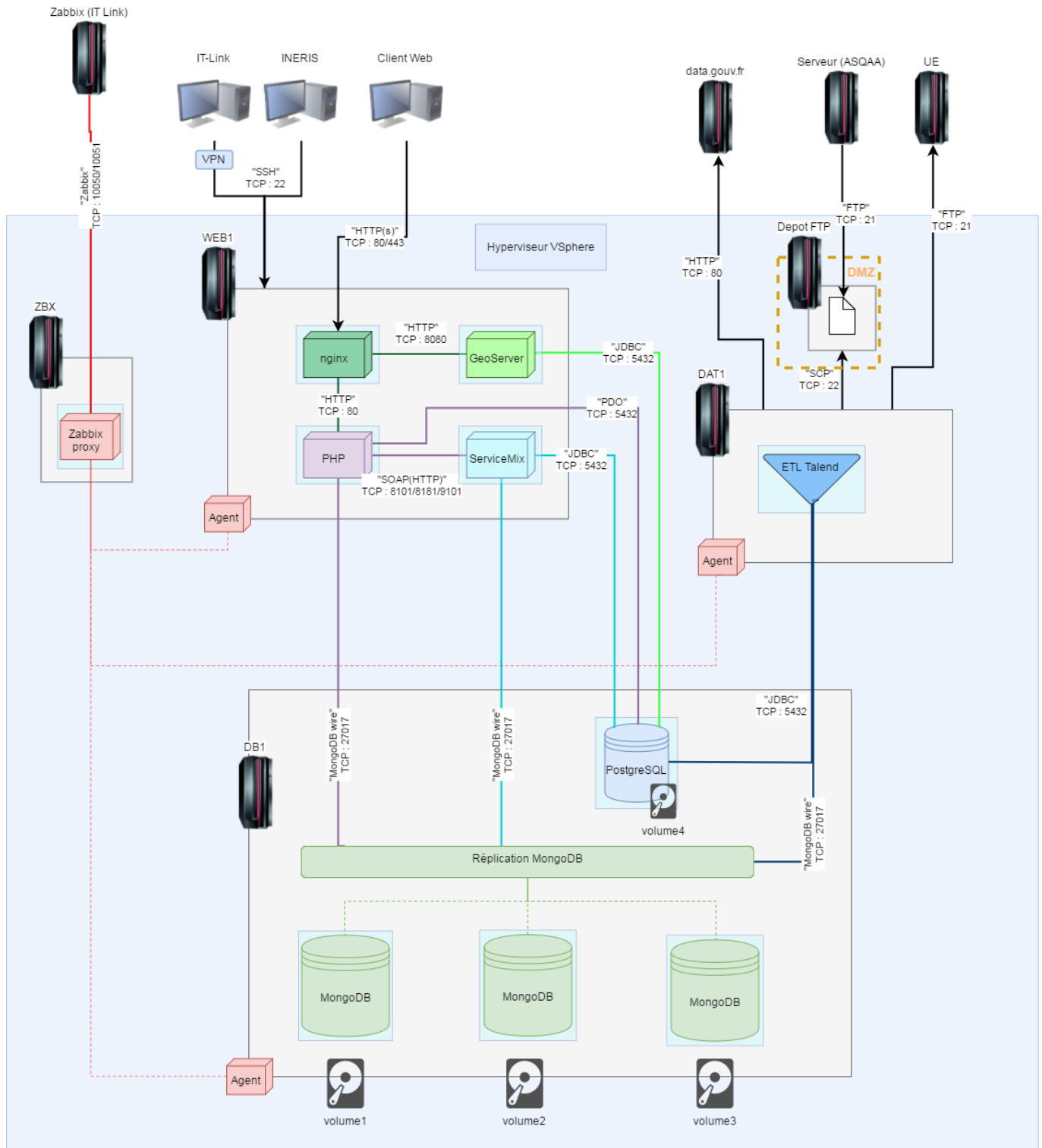


Figure 7 : Regroupement des bases de données sur la même VM.

Les flux mail ne sont pas représentés dans la figure ci-dessus mais seront dans le dossier d'architecture technique.

2.4.2. Dimensionnement des VM avec une Recette et une Production ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2. Les spécifications sont sommées pour conserver le même niveau de performance.

VM	CPU (#cores)	RAM (Go)	Disque HDD (Go)
WEB1	6	24	50
DAT1	20	32	200
ZBX	1	2	10
DB1	14	100	1900
Total	41	158	2160

2.4.3. Dimensionnement des VM avec une Recette et une Production non ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2.

Tableau 12 : Dimensionnement des VMs

VM	CPU (#cœurs)	RAM (Go)	Disque Production (Go)	Disque Recette (Go)
WEB1	6	24	50	50
DAT1	20	32	200	200
ZBX	1	2	10	10
DB1	14	100	1900	700
Total	41	158	2160	960

2.5. Scénario d'évolution 4

Ce scénario présente une solution intermédiaire entre le scénario 2 et 3, cherchant à équilibrer le nombre de VMs avec la disponibilité.

2.5.1. Ajustement de l'architecture

Cette architecture permet de séparer un des nœuds mongoDB et la base PostgreSQL, pour une disponibilité légèrement améliorée, tout en conservant une architecture compacte d'un point de vue VMs.

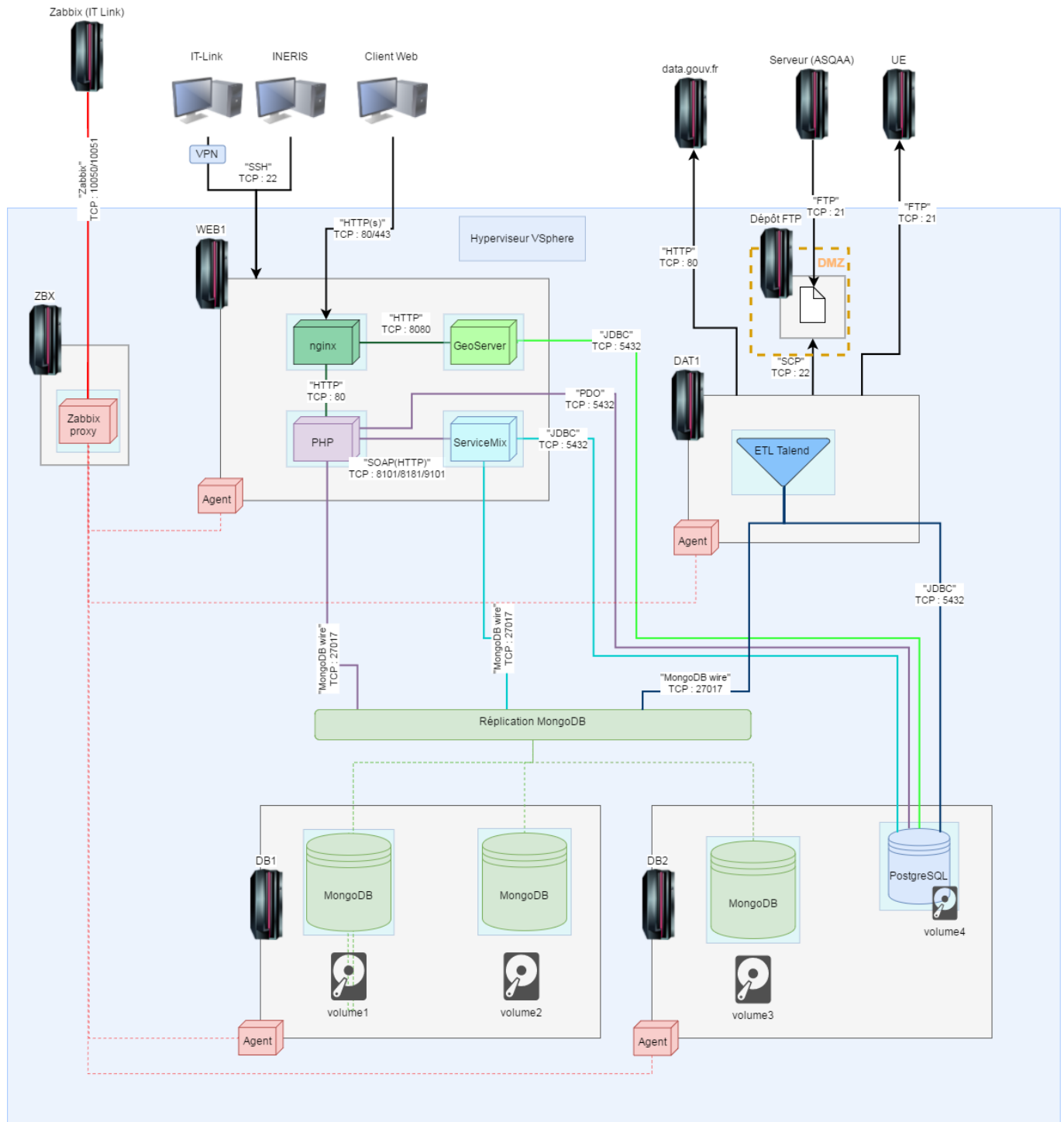


Figure 8 : Regroupement des bases de données sur 2 VMs différentes.

Les flux mail ne sont pas représentés dans la figure ci-dessus mais seront dans le dossier d'architecture technique.

2.5.2. Dimensionnement des VM avec une Recette et une Production ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2. Les spécifications sont sommées pour conserver le même niveau de performance.

VM	CPU (#cores)	RAM (Go)	Disque HDD (Go)
v	6	24	50
DAT1	20	32	200
ZBX	1	2	10
DB1	8	64	1200
DB2	6	36	700
Total	41	158	2160

2.5.3. Dimensionnement des VM avec une Recette et une Production non ISO

Le raisonnement du dimensionnement est le même qu'en 2.2.2.

Tableau 13 : Dimensionnement des VMs

VM	CPU (#cœurs)	RAM (Go)	Disque Production (Go)	Disque Recette (Go)
WEB1	6	24	50	50
DAT1	20	32	200	200
ZBX	1	2	10	10
DB1	8	64	1200	400
DB2	6	36	700	300
Total	41	158	2160	960

2.6. Alternative aux scénarios

Les alternatives présentées sont en option pour chaque scénario et cumulables entre elles.

2.6.1. Alternative mongoDB

Dans le but de réduire au maximum les besoins matériels, il est possible de supprimer une des instances mongoDB. Cette instance peut être remplacée par un « arbitre » qui ne recopie pas

les données, mais permet de compléter le besoin minimal de 3 instances du replicaSet. L'arbiter peut être déployé sur n'importe quelle VM autre que celles du nœud primaire et secondaire.

En revanche, cela a pour effet de réduire considérablement la disponibilité de mongoDB. En effet, lors d'un évènement (maintenance, panne...) sur un nœud, il ne reste plus qu'un seul nœud pour assurer le service. Cette solution est généralement déconseillée en production de ce fait.

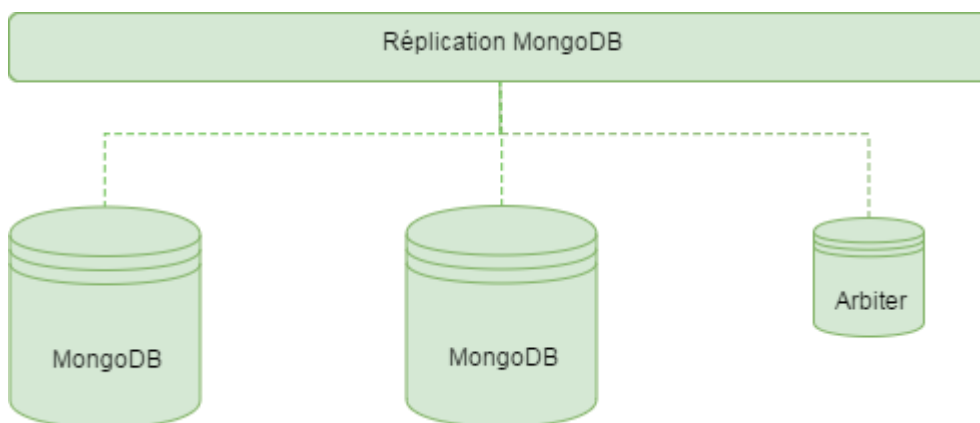


Figure 9 : Ajout d'un arbiter mongoDB pour conserver un replicaSet.

2.6.2. Alternative FTP

Le protocole actuel de dépôt de fichier ISO est FTP. Ne pouvant être changé immédiatement (contrainte ASQAA), les solutions proposées sont les suivantes :

- Identique à l'actuel :
 - Filtrage d'IP (sécurité identique à l'actuel, sécurité faible)
- Isolation de l'ETL dans le reste du réseau :
 - Filtrage par IP + configuration réseau complexe (sécurité moyenne)
 - Limitation pour les connexions aux bases de données
- VM de dépôt hors du réseau commun :
 - VM FTP dédiée (sécurité optimale)
 - L'ETL vient récupérer de lui-même les fichiers présents sur ce dépôt

2.7. Bilan des solutions proposées

Les montées en version des middlewares simples sont incluses dans le coût du scénario. Pour les middlewares avec une complexité forte, voir ci-dessous :

Scénario	Avantages	Désavantages	Nb VMs
----------	-----------	--------------	--------

1	<ul style="list-style-type: none"> Le moins de changement vis-à-vis de l'architecture actuelle. Séparation des services par VM (si DB3 tombe, PostgreSQL est encore accessible). 	<ul style="list-style-type: none"> Plus de VM donc plus coûteux. Certaines versions anciennes de middleware seront installées directement sur les VM. La mise à jour/rollback des middlewares implique une plus longue indisponibilité. Mise en conformité de GEOD'AIR dans l'environnement de développement ET dans la VM 	8
2	<ul style="list-style-type: none"> Une VM en moins, la séparation des services est faite par conteneurs. Le déploiement de nouvelles versions est facilité par les images Docker. La gestion des services (redémarrage, scalabilité...) peut-être automatisée avec Kubernetes. Mise en conformité de GEOD'AIR uniquement d'en l'environnement de développement 	<ul style="list-style-type: none"> Si une VM est indisponible, tous les services de la VM seront indisponibles. 	7
3	<ul style="list-style-type: none"> Avantages similaires au scénario 2, en utilisant un minimum de VMs. 	<ul style="list-style-type: none"> La VM DB1 mutualise les bases de données, si la VM n'est plus disponible, aucun accès en base n'est possible et GEOD'AIR est totalement muet. 	4
4	<ul style="list-style-type: none"> Avantages similaires au scénario 2, en utilisant un nombre moyen de VMs. 	<ul style="list-style-type: none"> Un nœud mongoDB partage le disque de PostgreSQL. 	5

La montée vers la version 7 de PHP peut s'effectuer dans une phase postérieure à la migration de l'environnement. La montée en version de PHP implique une étude pour affiner la charge de réalisation.

2.7.1. Avantages et désavantages de certaines évolutions

Ci-dessous une liste d'idées et d'axes d'amélioration de l'application :

Solution	Avantages	Désavantages
Remplacer mongoDB par PostgreSQL	<ul style="list-style-type: none"> • Plus simple à utiliser et maintenir. • Plus rapide sur certains types de requêtes. 	<ul style="list-style-type: none"> • Très coûteux de recoder les interfaces à mongoDB.
Remplacer ServiceMix par un serveur java plus classique	<ul style="list-style-type: none"> • Plus grande maîtrise des services. 	<ul style="list-style-type: none"> • Très coûteux de coder un nouveau serveur back-end.
Implémenter une solution de failover pour les différents services	<ul style="list-style-type: none"> • Plus haute disponibilité. 	<ul style="list-style-type: none"> • Potentiellement coûteux selon la méthode utilisée (VMs supplémentaires, installation/config Kubernetes)

3. Organisation de l'exploitation

3.1. Rôles de SPIE

La partie ci-dessous est un extrait du document des niveaux de services SPIE.

SPIE assure un module de service A2 pour l'application GEOD'AIR, ce qui implique les responsabilités suivantes :

- Supervision des infrastructures et des systèmes
 - Une couverture continue de service « Standard » sans interruption pendant les horaires définis par le SLA en opérant :
 - La supervision des couches infrastructures (Matériel et OS),
 - La supervision des serveurs virtuels (OS, services, processus, etc.),
 - La supervision de la capacité.
 - L'application des procédures et des consignes prédéfinies et validées avec INERIS
 - Une traçabilité de tous les événements anormaux et des interventions correspondantes.
- Exploitation des infrastructures et des systèmes
 - La gestion des sauvegardes
 - L'application de la politique de sécurité
 - La gestion des incidents de production (pendant les horaires définis par le SLA)
 - La gestion de documents d'exploitation et la production des indicateurs
- Administration des infrastructures et des systèmes
 - La résolution des incidents et problèmes escaladés par les équipes de niveau 1 ou la supervision
 - L'analyse des différents logs systèmes
 - La création et le maintien en condition opérationnelle de la documentation technique nécessaire à l'industrialisation et au bon fonctionnement des équipes de niveau 1
 - La réalisation de script pour automatiser et industrialiser les tâches récurrentes
 - L'ouverture si nécessaire et l'exécution des changements normaux ou mineurs préalablement validés

3.2. Rôles d'IT Link

IT Link assure les responsabilités suivantes :

- Supervision des middlewares
 - Supervision des métriques middlewares et OS à travers Zabbix
- Exploitation des middlewares
 - La gestion des incidents de production et de recette

- TMA de l'application
 - Maintenance corrective
 - Lots d'évolutions
 - Assistance et conseils

4. Ébauche du plan de migration

- Mise en conformité de GEOD'AIR
- Récupération des dumps et des fichiers non versionnés de production
- Déploiement de la solution choisie :
 - Scénario 1 :
 - Installation des middlewares sur les VMs
 - Installation des exécutables et des scripts
 - Restauration des bases de données
 - Scénarios 2 et 3 :
 - Installation de Docker et déploiement des conteneurs sur les VMs
 - Configuration et paramétrage du partage des ressources Docker si besoin
 - Restauration des bases de données
- Vérification des connexions entre les services, avec l'aide de la matrice des flux
- Réintégration des flux de données :
 - Dépôt de fichier sur le serveur FTP
 - Consommation du fichier par l'ETL
 - Intégration du fichier en base
 - Calcul des statistiques
 - Envoi des rapports vers data.gouv.fr et l'UE
 - Envoi de mail
- Test du dépôt de fichier par les ASQAA
- Recettage du site à travers l'IHM pour vérifier le bon fonctionnement :
 - Consultation
 - Export
 - Génération de rapports
 - Envoi de mail