Rôles / Services

Notions théoriques liées aux services et divers produits sur cette couche.

- Active Directory - Méthode AGDLP
- Docker - Introduction à la "containerisation"
- Filer - DFS & DFSR

Active Directory - Méthode AGDLP

Difficulté : Débutant

Notions : Active directory, Organisation, Bonne pratique, méthodologie AGDLP.

I. Introduction

La méthode AGDLP (**A**ccount **G**lobal **D**omain **L**ocal **P**ermission) est une méthode de gestion de droits et permissions à travers les groupes Active Directory et plus largement LDAP.

Cette méthode poursuit un objectif triple :

- **Management** : Cela facilite la délégation des accès et de la gestion de ceux-ci. Le fait de dé-corréler l'utilisateur de ses droits d'accès rends la solution scalable et facilement réversible.

- **Audit / Respect RGPD** : Elle aide également à la conformité RGPD car grâce à elle, il deviens facile et rapide de savoir avec précision qui a accès à quoi. Cela rends la traçabilité plus simple et précise.

- **Cyber sécurité / Moindre privilège** : Elle permet enfin de s'assurer une normalisation des droits d'accès en s'assurant que chaque utilisateur dispose uniquement des droits strictement nécessaires et que la portée de ces droits est correctement ajustée.

La combinaison de ces trois buts résulte en une gestion qui a priori peut sembler complexe à la mise en oeuvre, mais qui se révélera par la suite bien plus simple, transparente et efficiente dans son utilisation courante.

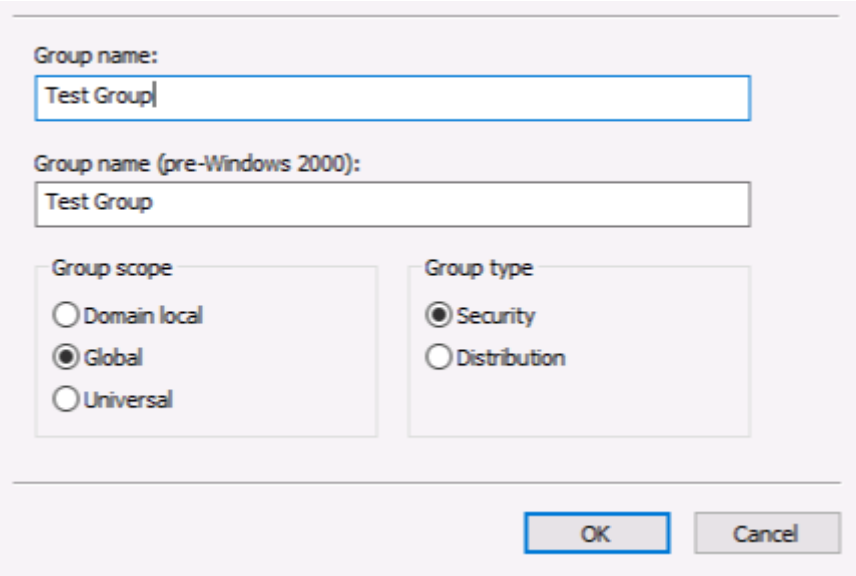
Dans un second temps, il deviendra aussi possible d'encadrer correctement la délégation de la gestion des accès et des privilèges.

Cette méthode peux s'appliquer indépendamment aux serveurs de fichiers, aux différents services et applications de l'entreprise, ainsi que (par le biais du SSO) aux rôles des application fédérées.

> **Info** : Cette méthode prends tout son sens dans un environnement multi-domaines comme c'est de plus en plus fréquent d'en voir avec la logique de tiering.

II. Rappel : Portée des groupes AD

Avant de rentrer plus en détail sur l'application de cette méthode, il est nécessaire de rappeler certaines bases du fonctionnement des groupes Active Directory (ou LDAP) car c'est sur eux que va intégralement s'appuyer cette méthode.



Pour rappel, il exite deux types de groupes :

- **Security** (sécurité) : Ce type de groupe est celui qui va être utilisé dans ce contexte. C'est celui qui permet de gérer les permissions et les droits d'accès ainsi que l'application des GPO.
- **Distribution** : Ce type de groupe sert à gérer les listes de distribution pour les mails.

A cela s'ajoute la portée du groupe. c'est ce point qui va être central dans l'utilisation d'AGDLP. Il s'agit du niveau de visibilité du groupe.

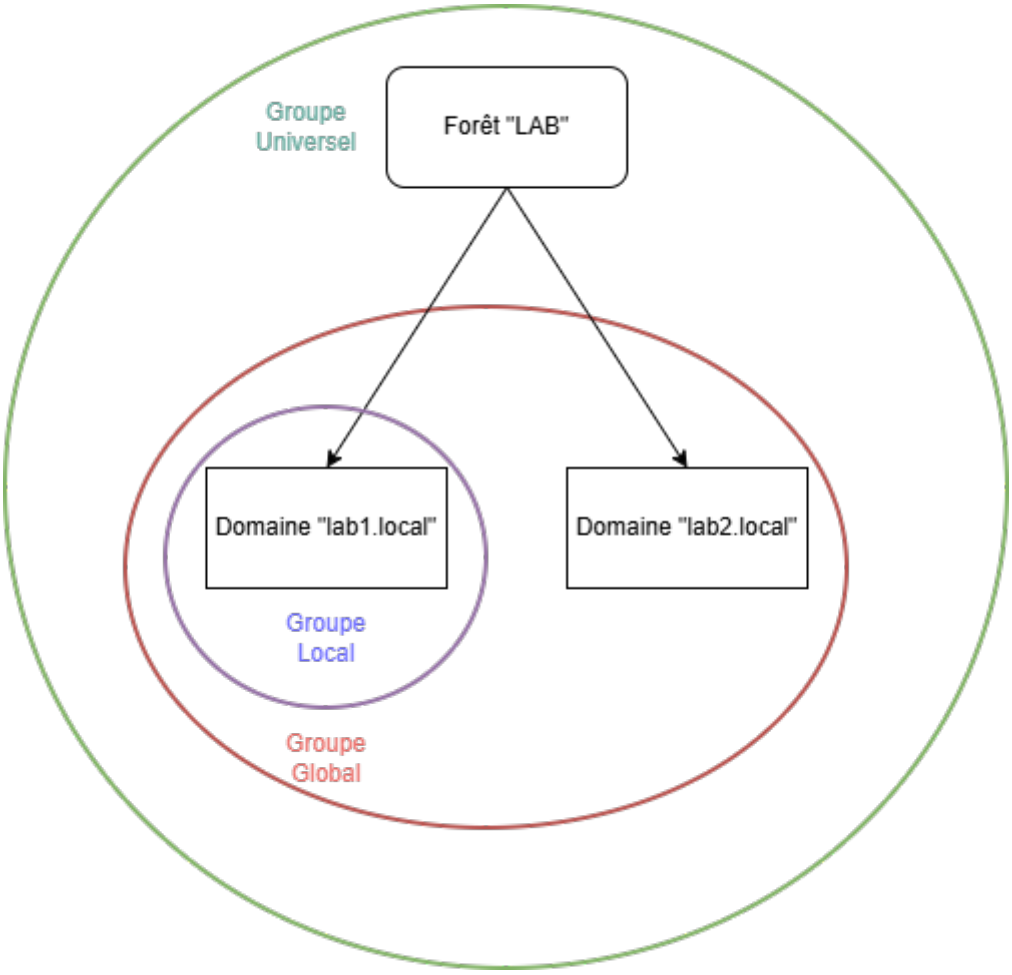
Il y a 3 niveaux de portée :

- **Domain Local** : Le groupe existe dans le domaine local. Mais n'est pas visible depuis les autres domaines.
  - Ils peuvent contenir : des utilisateurs du domaine, des groupes globaux d'autres domaines, des groupes universels.

- **Global** : Le groupe existe dans le domaine local et est visible depuis les autres domaines de la fôret.
  - Ils peuvent contenir : des utilisateurs du même domaine, des groupes globaux du même domaine.
  - Ils peuvent être membre : de groupes locaux dans n'importe quel domaine, de groupes globaux du même domaine, de groupes universel.

- **Universel** : Le groupe existe au niveau de la forêt active directory est est visible partout.
  - Ils peuvent contenir : utilisateurs de n'importe quel domaines, groupes globaux de n'importe quel domaine, autres groupes universels.

- Ils peuvent être membre : de groupes locaux dans n'importe quels domaines, d'autres groupes universels.



III. Fonctionnement de la méthode

3.1 Principe général

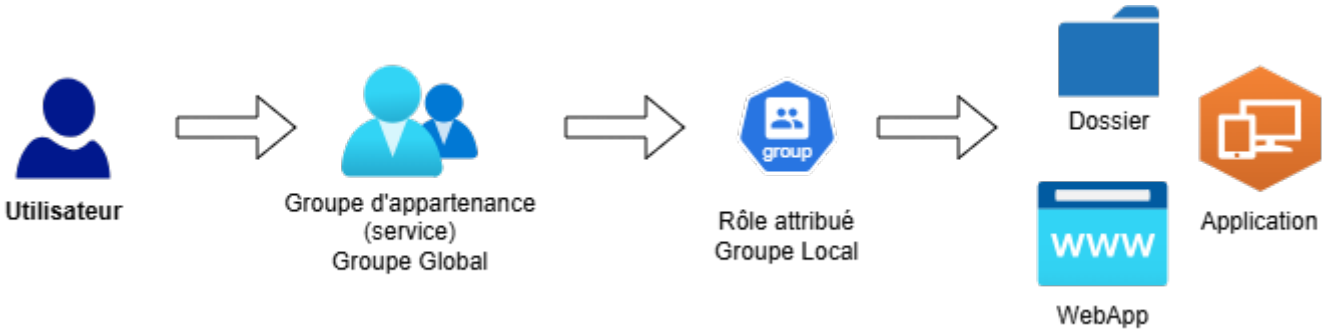
Comme vu ci-dessus, les groupes peuvent être imbriqués les uns dans les autres et c'est ce principe là qui va nous intéresser dans l'application de la méthode AGDLP.

L'idée de cette méthode est de dissocier les notions de **rôles** (droits effectifs attribué à un utilisateur) et de **groupes** d'appartenance.

Afin de limiter également la portée des droits, il faudra utiliser pour ces rôles des groupes avec une portée minimaliste.

Puis pour octroyer un rôle à un groupe d'utilisateur, il faudra ajouter le groupe des utilisateurs dans le groupe rôle.

Ainsi, si un utilisateur intègre un service dans l'entreprise, son rajout dans le groupe du service lui conférera automatiquement les accréditations (rôles) inhérents à ce service.



3.2 Normalisation

Afin de pouvoir se retrouver dans les noms et types de groupes, il va falloir adopter une normalisation stricte et une convention de nommage claire.

Les bonnes pratiques observées tendent vers la convention suivante :

- Le préfixe ***GRP*** suivi du séparateur _
- Le type de groupe (sécurité, GMSA, liste, ...)
- Un caractère de séparation. Ici _
- La ***portée*** du groupe (local, global, universel).
- Un caractère de séparation. Ici _
- La finalité de l'***utilisation*** du groupe.
- Un séparateur différent. Ici **-**
- Le ***nom*** du groupe (ou du dossier pour les groupes de sécurité).

> **Info :** les ***groupes de sécurités*** pour l'accès aux dossiers seront suffixé par ***_RO*** (read only), ***_RW*** (read-write) ou ***_FC*** (full control).

Préfixe	Type	Séparateur	Portée	Séparateur	Application	Séparateur	Usage
GRP_	SEC	_	GL	_	GPO_EXL	-	PrintServers
	GMSA		GG		GPO-INC		RDSSessionHosts
	LST		UN		RDS		Farm1
					APP		AppName
					VPN		Admins, Users
					PROXY		RestrictedUsers, Noaccess, ExtendedAccess
					VCSA		LAB1, LAB2
					FIL		Comptabilité_RO
					...		

> **Conseil** : les ***listes de distributions*** seront simplement préfixées avec '***&***' suivi du nom de la liste

GRP-GLO-RDS_AllUsers

GRP-GLO-APP_Sage

GRP-LOC-ROXY_NoInternetAccess

&ServiceComptabilité

IV. Mise en application

4.1 Appliqué au systèmes de fichiers

Exemple : Dans le cadre de son service comptabilité, l'entreprise dispose d'un partage ***//share/comptabilité***.

L'entreprise veux mettre en place 3 niveaux d'accréditation :

- Les comptables gèrent le service et doivent avoir accès en contrôle total au dossier partagé.
- Les assistants n'ont pas besoin de créer ou supprimer des choses, mais doivent pouvoir réaliser des modification dans les fichiers existants.
- Les auditeurs sont des personnes qui viennent auditer les comptes et doivent avoir un regard sur les pièces comptable, mais non pas besoin d'écrire.

Dans ce cas de figure, la première étape consiste à créer le répertoire et les groupes d'autorisations (rôles). En groupe local, car les droits ne s'appliquent que dans ce domaine.

GRP_SEC_GL_FIL_Compta_FC

GRP_SEC_GL_FIL_Compta_RW

GRP_SEC_GL_FIL_Compta_RO

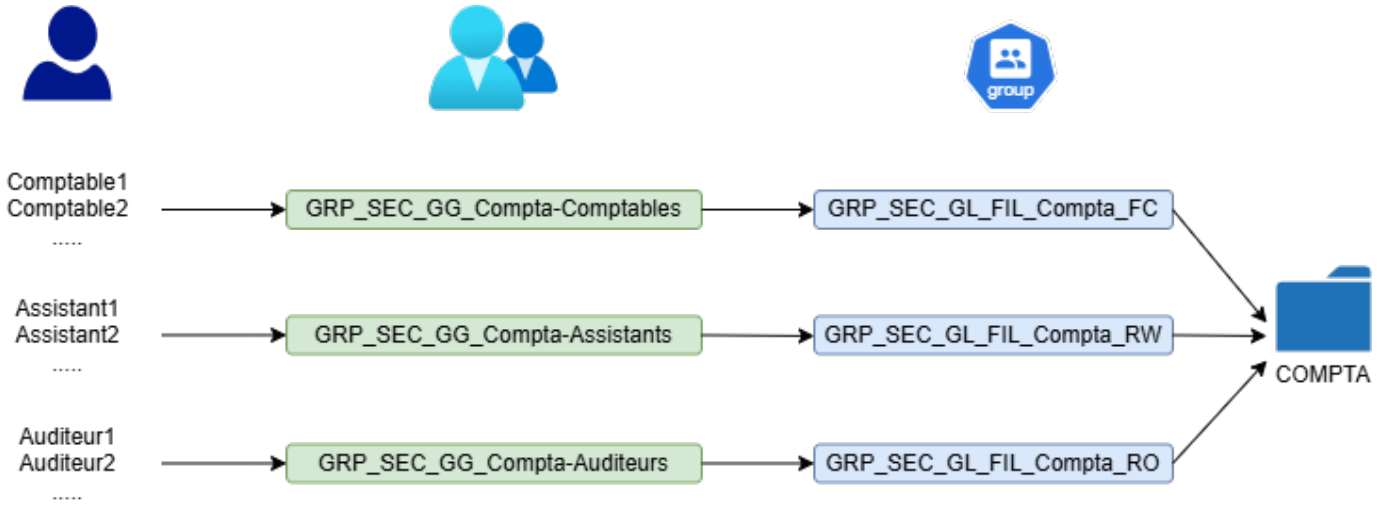
Puis créer les groupes globaux contenant les utilisateurs et les intégrer dans les groupes locaux.

GRP_SEC_GG_Compta-Comptables

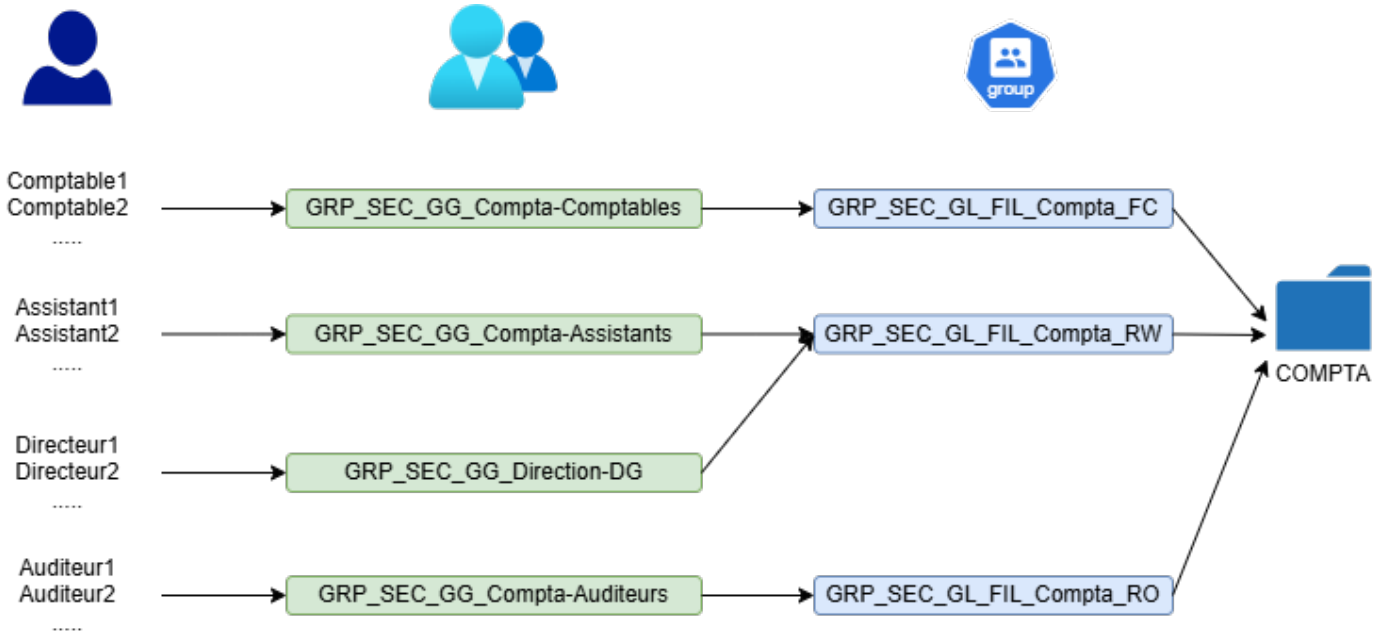
GRP_SEC_GG_Compta-Assistants

GRP_SEC_GG_Compta-Auditeurs

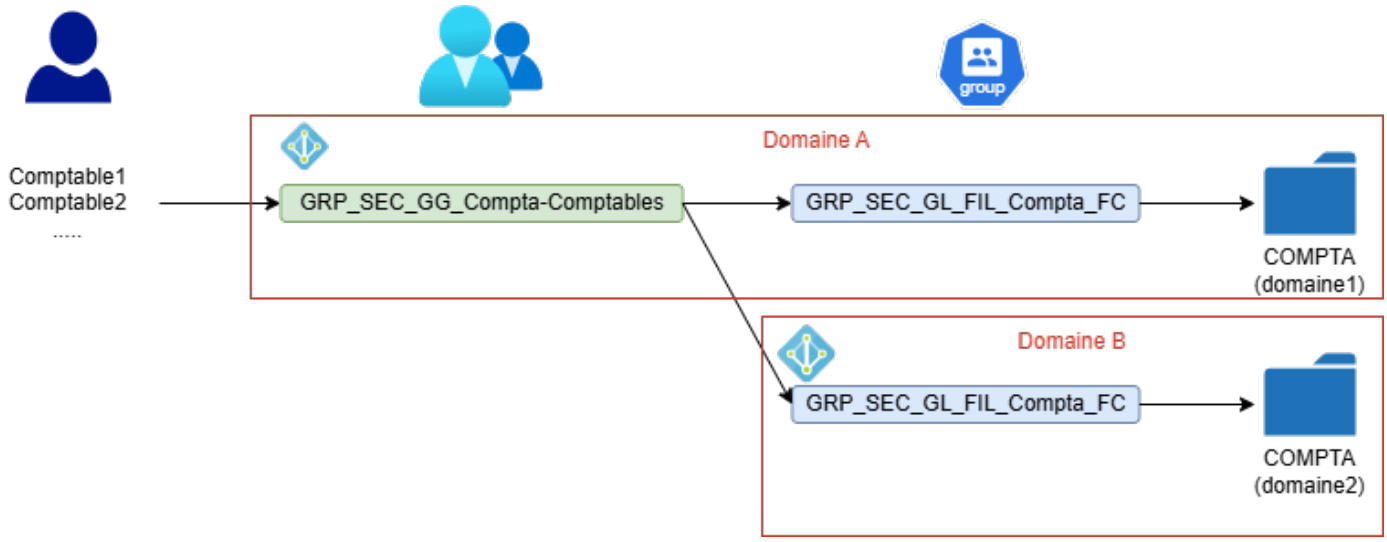
Enfin, il suffira d'ajouter / Retirer au besoin les utilisateurs dans les groupes :



Si l'on veut par exemple que le groupe Direction contenant le PDG et les membres du conseil ai également un droit de regard et de modification sur les fichiers de la comptabilité, il faudra ajouter cela de la façon suivante.



A l'inverse, si un même groupe, par exemple de comptables du siège social doit avoir accès aux dossiers comptabilité des autres domaines. Les groupes globaux étant visibles depuis les autres domaines, il faudra l'ajouter dans les groupes locaux sur les autres domaines.



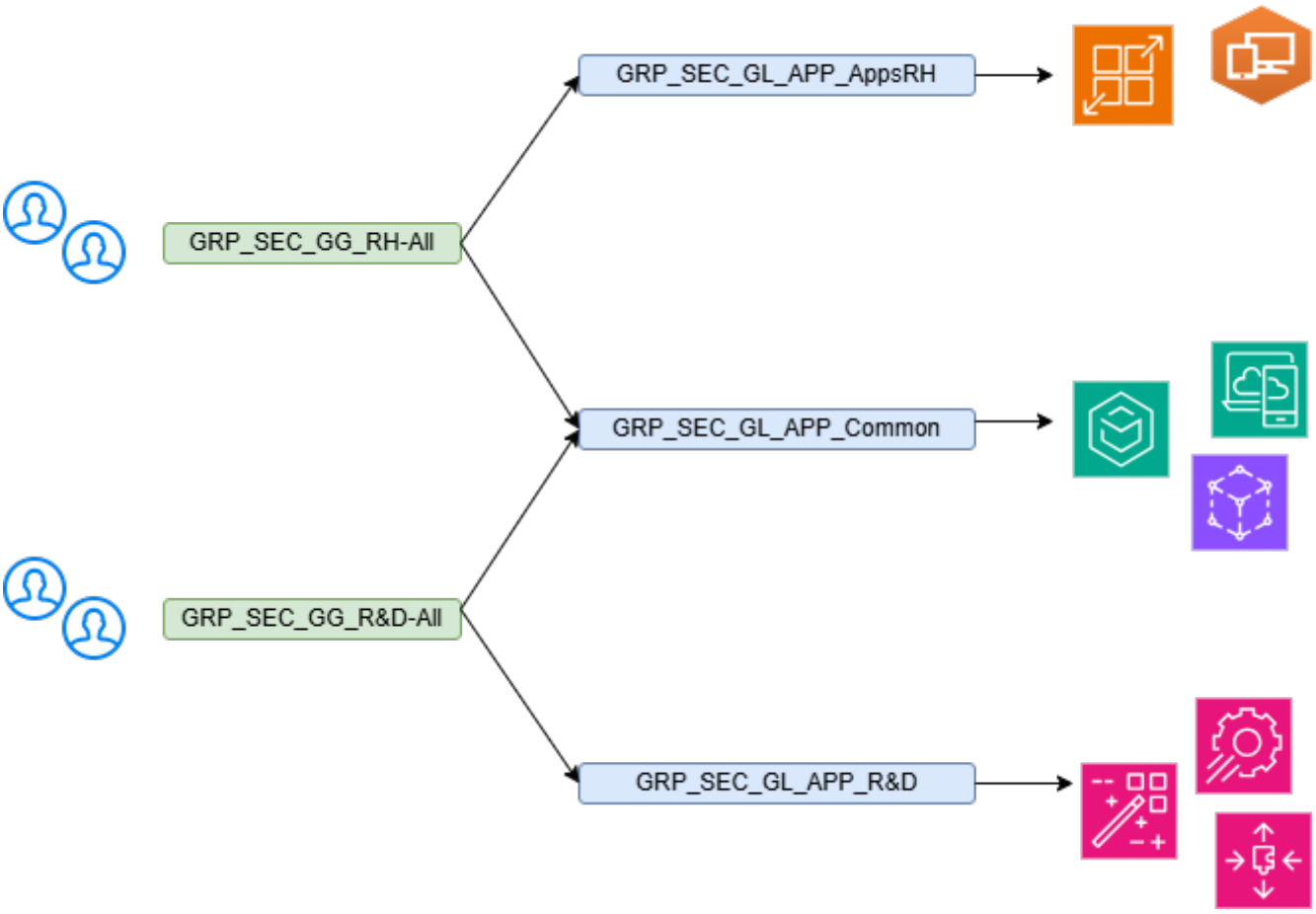
4.2 Appliqué aux applicatifs & services

Il est possible de poursuivre cette logique sur les applicatifs et services. Par exemple dans le cadre d'une ferme RDS, plusieurs applications sont partagées dans différentes collections.

Par exemple :

- le service R&D utilise un groupe d'application de conception de produits électronique en RDP.
- Le service RH utilise des application pour la gestion des bages, etc...
- Les deux services accèdent également à un pool d'applications communes à tout le personnel.

La définition se fera comme suit :



> **Info** : De même que précédemment, les groupes globaux pourront être ajouté dans des groupes locaux d'autres domaines si nécessaires. Cette méthode peut aussi s'appliquer par exemple pour des services comme le proxy, le VPN, etc...

4.3 Appliqué au SSO et aux rôles

Dans la plupart des applications modernes, la notion de rôle est déjà intégrée au sein de l'application et ce rôle peut être appliquée directement à un groupe ou utilisateur.

Ainsi il pourra être mis en place le même principe. Pour attribuer ces rôles automatiquement, cela pourra se faire de deux manières :

- Soit à travers le SSO directement qui mappera l'appartenance au groupe au rôle correspondant.
- Soit directement dans l'application, auquel cas le système reste le même que précédemment.

Le système SSO intégrera alors dans le jeton d'authentification les appartenances au groupes, ce qui permettra d'effectuer les claims (demande de rôles).

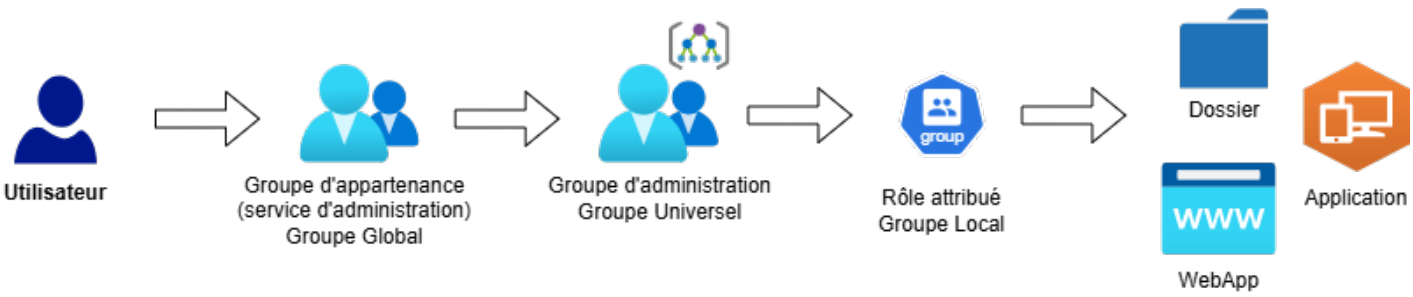
V. Variantes du modèle

5.1 AGUDLP (grosses structures)

Il s'agit de la même méthode, mais incluant également les groupes universels pour gérer les approbation sur l'ensemble des domaines.

Adapté dans un environnement multi-domaine, ou de tiering au sein d'une forêt, elle permet d'avoir par exemple un domaine d'administration et plusieurs domaines clients.

Il faudra alors ajouter les administrateurs dans un groupe global du domaine d'admin, puis ces groupes d'administration seront intégrés dans un groupe universel afin d'en faciliter l'intégration au sein des différents domaines.



5.2 AGLP

Cette variante consiste à utiliser des groupes locaux sur la machine directement au lieu d'un groupe active directory.

Cela peut être nécessaire sur des applications utilisant des groupes locaux des machines ou certains services créant eux-même des groupes locaux sur les machines afin de gérer l'accès aux ressources du services.

Ou encore certaines ressources et services sur linux.

> **Info** : Il existe également d'autres vairantes comme AGUDL P, AGUGDLP, mais celles-ci sont rare et ne constituent pas la norme de l'utilisation.

VI. Bonnes pratiques

Afin de garantir une efficacité optimale de cette méthode, il faut considérer un certain nombre de bonnes pratiques associées.

A commencer par le fait de l'intégrer dans un plan plus large. Cela va de pair avec :

- Une convention de nommage claire et explicite (comme vu ci-dessus).
- Une documentation des niveaux accréditations et une cartographie des accès.
- Une politique d'audits réguliers des groupes et permission afin de s'assurer de leur conformité.
- Mettre en place une politique de gestion du cycle de vie des groupes (validation des ajouts / suppression des membres).
- Eviter les imbrication inutiles et limiter au minimum celles-ci.
- Contrôler les délégations sur la gestion de ces groupes.

VII. Erreurs à éviter

De même que les bonnes pratiques garantissent un fonctionnement optimal, de mauvaises pratiques peuvent également venir saper l'efficacité de la méthode :

- Donner des permissions directement aux utilisateurs (bypass de la chaîne d'autorisations).
- Utiliser des groupes globaux sur les ACL (contraire au principe de moindre privilège).
- Mélanger rôles, permissions dans un même groupe.

- Ne pas nettoyer les groupes, utilisateurs, permissions obsolètes.
- Avoir une mauvaise gestion des comptes de services.

VIII. Conclusion

Si cette méthode peut ne pas être intuitive voire pénible à mettre en place, elle facilitera grandement la gestion courante des accès et des droits. Elle permettra également, sous réserve de respect des bonnes pratiques et d'éviter les erreurs, de permettre un audit rapide et efficace des accès effectifs pour un utilisateur ou un service de l'entreprise.

> **Conseil** : De plus, sa mise en oeuvre et son maintient peuvent être grandement améliorées et facilitées par la mise en place de systèmes d'automatisation à travers des scripts ou des catalogues de services IaC. Réduisant ainsi considérablement la marge d'erreur humaine et garantissant la conformité.

Docker - Introduction à la "containerisation"

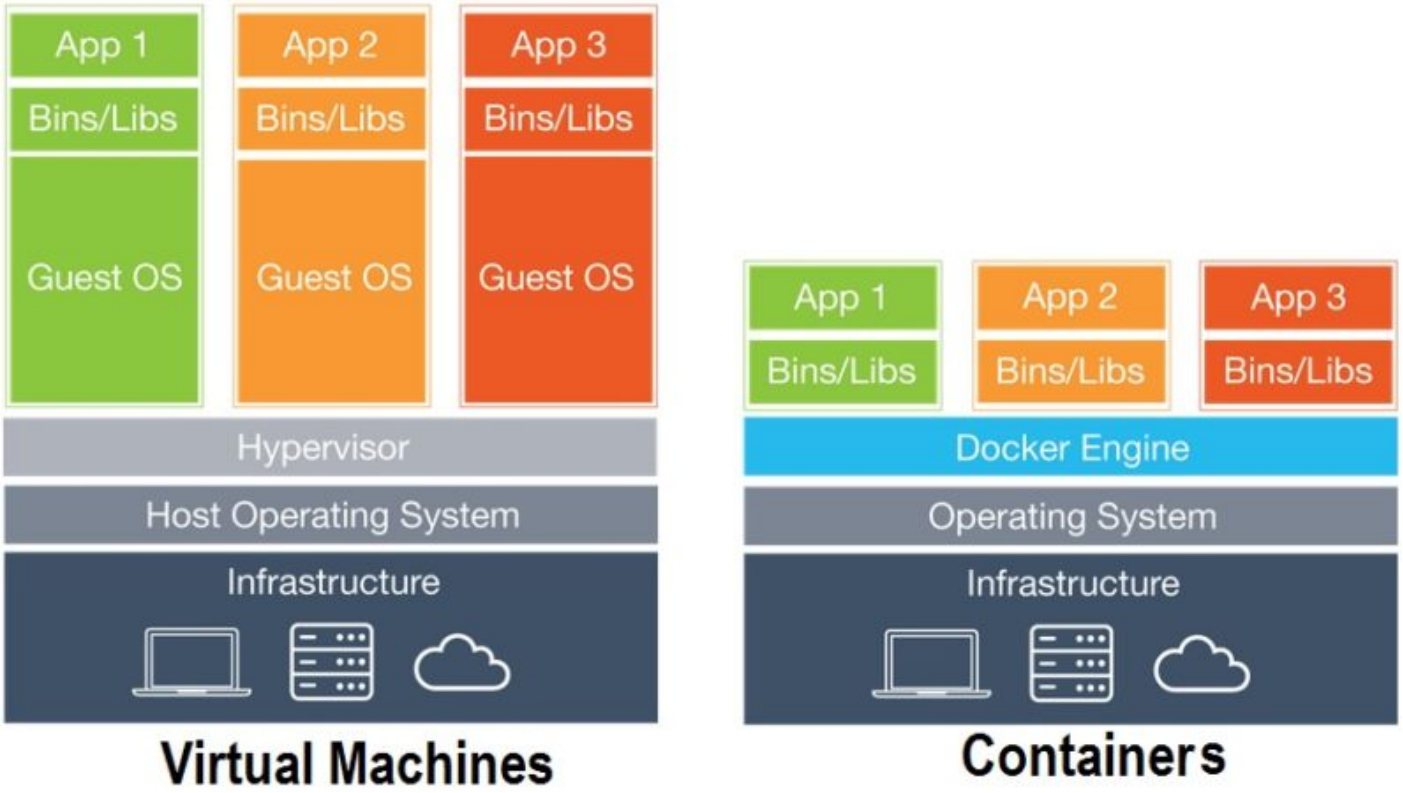
Difficulté : Intermédiaire

Notions : Containers, systèmes de containers, docker.

I. Introduction

Tout comme la virtualisation avais permis un bond en avant en permettant de placer plusieurs machines virtuelles sur un hôte physique, la containérisation va plus loin en ne virtualisant plus un système entier mais en cloisonnant directement une couche applicative ou un service dans un container.

Basé sur un système d'image, celui-ci permet de virtualiser un morceau de système sur lequel viendront s'appuyer les containers contenant les ressources (binaires, configurations, librairies et dépendances) et auquel il sera possible de connecter des volumes persistant contenant les données.



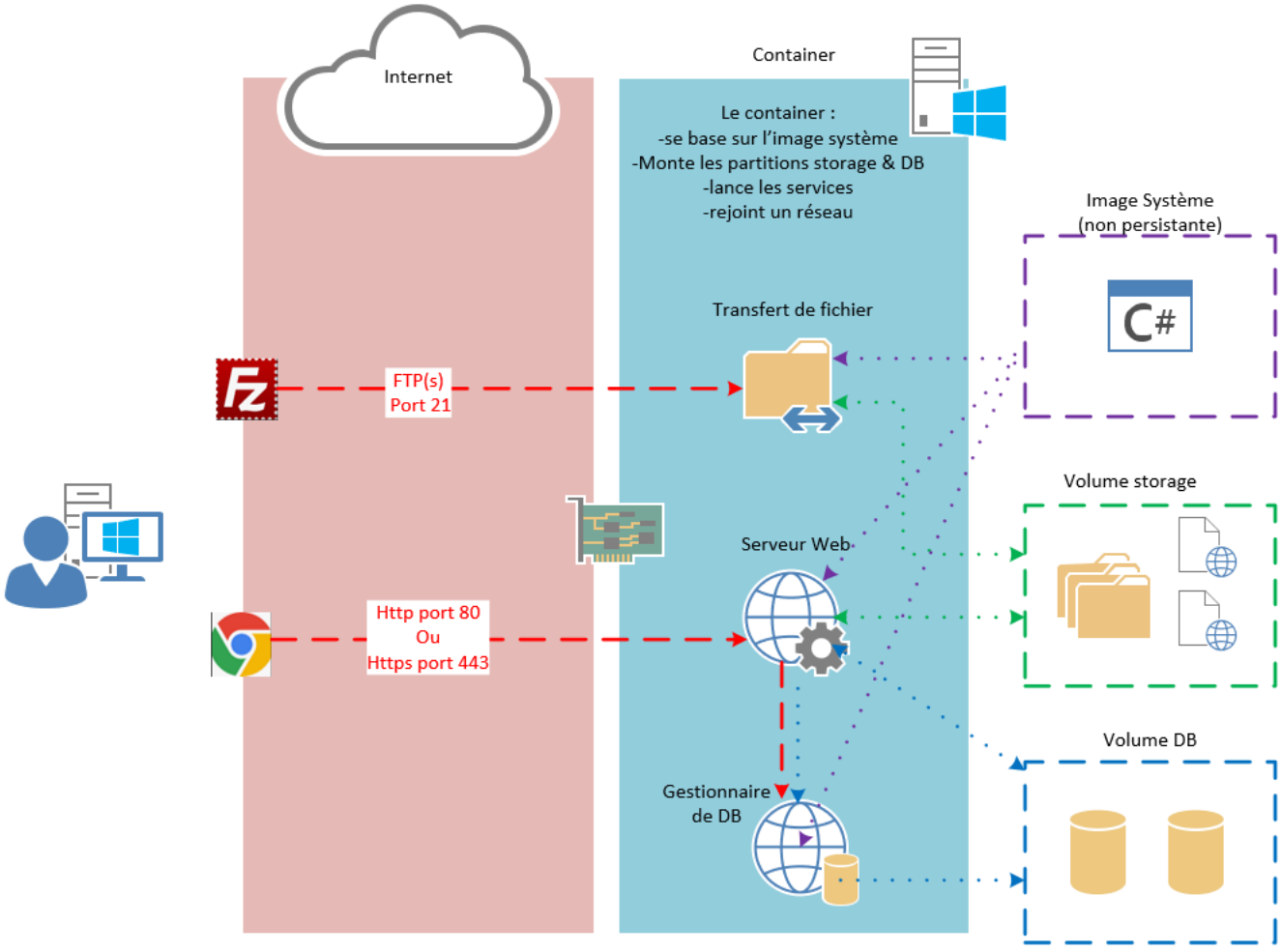
Cela est une excellente alternative en terme de performance, la mutualisation des images permettant un gain significatif du fait de la non nécessité de virtualiser une couche hardware complète dans la plupart des cas.

Mais également en terme de gestion et de déploiement. Une fois les images créées, elle sont déployables en quelques minutes et utilisables immédiatement. De plus si plusieurs containers utilisent la même image et nécessitent un mise à jour, il sera simple de mettre à jour l'image et la pousser en production. les containers pourront ensuite être liés a cette nouvelle image, réduisant le temps de mise à jour et les indisponibilités.

Enfin en terme de sécurité. Chaque application utilisant son propre environnement cloisonné, il sera très difficile d'infecter les systèmes adjacents et la compromission d'un service aura des effets uniquement sur celui-ci.

De plus, les images étant non persistantes, un rechargement de celle-ci supprimera les éventuelles traces d'attaque et permettra de revenir rapidement à une configuration sûre.

Cela permet également de personnaliser grandement l'architecture des services. Par exemple voici un container web :



Enfin Le dernier avantage et non le moindre, est la grande capacité d'automatisation de l'outil.

Les images et containers, ainsi que des architectures complètes peuvent être créées a partir de fichiers d'instructions (la plupart en YAML) qui seront utilisés comme base à travers des API.

Ce qui permet de faire de "l'infrastructure as code" et de publier en quelques étapes des services voire des infrastructures complètes architecturées en amont.

II. Principes de bases

Les images : Les images sont des versions packagées de services. Elles sont construite à l'aide de fichiers que l'on appelle 'dockerfiles'. Le plus souvent, elle partent d'une couche OS sur laquelle le(s) service(s) et le contenu seront préconfigurés et packagés puis fournis sous forme d'image pour un déploiement rapide et une grande souplesse dans la gestion des containers utilisant ces

images. Lors de l'utilisation par un container, le contenu de l'image est dit 'non persistant'. En effet, si le container est arrété ou recontruit, l'image redémarre sur son état initial.

Les volumes et points de montage : Du fait de la non persistance des images, il n'est pas possible d'y stocker des données dites 'vivantes' dans le container. Pour cela, il faudra lui attacher des volumes ou des fichiers. Il s'agit des ressources qui seront partagées entre l'hôte et le container. Cela peut être des fichiers de configurations par exemple qui seront montés dans le container sur un chemin précis ou dans le cas des volumes. Lors de la création de ceux-ci, un espace dédié est créé sur l'hôte et sera utilisé ensuite dans le container sur un point de montage. L'on y placera toutes les données qui feront vivre le service (site web, base de donnée, fichiers du service, logs...)

Les containers : Le container est donc ce qui résulte de l'utilisation des éléments ci-dessus. Il va utiliser une image pour fournir un service et les volumes pour stocker la donnée.

Les services : Sur un cluster (swarm), les containeurs ne sont plus addressés directement en tant que tel mais font partie d'un service. Celui-ci permet de disposer de propriétés de déploiement comme de la haute dispo, des contraintes de placement sur des noeuds du cluster, le nombre de copies de containers, etc...

Les stacks : Une architecture rendant un service peut être constituées de plusieurs containers ou services. par exemple dans le cadre d'un hébergement web, il peut y avoir un container wordpress, puis un container DB et un container SSH, tous constitutifs du 'stack' hebergement.

Les réseaux : Les réseaux permettent à différents services de comuniquer entre eux. Par exemple permettre à un container web de se connecter à un container SQL, ou encore à un reverse proxy d'adresser un serveur web.

L'exposition des ports : Par défaut les containers sont dans des réseaux isolés ou bridgés non accessible de l'extérieur. Il faut pour cela natter des ports. Par exemple un port 8082 en TCP sur le port 80 d'un container. dans ce cas, c'est l'hôte qui devra être addressé avec le port natté (ici: 8082). On parkle alors d'une "exposition de port".

Les secrets : Certaines informations sont par nature sensibles. Les comptes ou mots de passes, des clés, des tokens, etc... Pour se passer ses informations entre les noeuds ou les monter dans un ou plusieurs container sans exposer ces informations il est possibles de créer des secrets. Ce sont des fichiers chiffrés qui sont directement généré à partir d'une saisie d'information protégées et ne seront ensuite utilisable qu'entre les membres du cluster et aux containers. Cela permet d'utiliser ces informations en toute sécurité.

Les dockerfiles : Ce sont des fichiers contenant une séquence d'instructions qui vont être utilisés pour générer une image. Cela sert dans le cadre de l'automatisation. Exemple :

```
FROM image_name #partir de l'image désignée (ou FROM scratch) pour recompiler une image en partant de zéro.

LABEL label1="valeur1" label2="valeur2" label3="valeur3" #définit les tags de l'image

VOLUME ['/chemin/vers/dossier'] # si pas de volume associé, par défaut monte un volume à cet endroit

ARG ARG1="value" #variable 1
ARG ARG2="value" #variable 2

WORKDIR /chemin/racine #chemin sur lequel on se place pour exécuter les commandes

COPY /chemin/vers/fichier /chemin/dans/container #copie un fichier présent sur la machine dans le container

RUN commande1 # lance une commande dans le container
RUN commande2
RUN commande3

EXPOSE 80/tcp #expose le port voulu dans le protocole voulu

ENTRYPOINT ["command", "arg1", "arg2"] #donne le processur ou service à observer pour le démarrage du container
```

> Pour plus d'informations : Dockerfile reference

Les fichiers compose : Les fichiers compose sont des fichier descriptif de l'architecture d'un service ou d'un stack. Ils permettent de générer les services/stacks à partir de ce fichier de manière automatisée. Cela rentre dans le cadre de "l'infrastructure as code".

Exemple de fichier compose pour un stack wordpress :

```
services:
  db:
    # We use a mariadb image which supports both amd64 & arm64 architecture
    image: mariadb:10.6.4-focal
    # If you really want to use MySQL, uncomment the following line
    #image: mysql:8.0.27
    command: '--default-authentication-plugin=mysql_native_password'
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
      - MYSQL_PASSWORD=wordpress
    expose:
      - 3306
      - 33060
  wordpress:
    image: wordpress:latest
    volumes:
      - wp_data:/var/www/html
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=wordpress
      - WORDPRESS_DB_PASSWORD=wordpress
      - WORDPRESS_DB_NAME=wordpress
volumes:
  db_data:
  wp_data:
```

Le swarm : le swarm est un cluster composant un ensemble de noeuds sous docker. Un swarm est initialisé par un noeud 'manager' et comportera ensuite un ensemble de 'worker'. Cela permettra de répartir des charges de travail et/ou de fiabiliser certains services en les répliquants sur plusieurs noeuds.

Filer - DFS & DFSR

Difficulté : Intermédiaire

Notions : Serveurs de fichiers, partages, Réplication, cluster.

I. Introduction

Le DFS (**D**istributed **F**ile **S**ystem), ou ***système de fichiers distribué*** en français, est un rôle serveur permettant de centraliser la gestion et la publication des partages sur différents serveurs de fichiers en s'appuyant sur les espaces de noms.

Le principe est d'avoir un ou plusieurs serveurs gestionnaires de ces espaces de noms qui vont créer une racine 'virtuelle' au sein du domaine depuis laquelle tous les partages seront accessibles. Au niveau de l'utilisateur, ce chemin se substitue au chemin réel vers le dossier partagé.

Cela permet de dé-corréler l'accès aux fichiers du point de vue utilisateur de l'emplacement réel des données.

Cette dé-corrélation entraîne ainsi plusieurs avantages :

- Possibilité d'ajouter de la redondance sur les sources de partages
- Possibilité de stocker différents partages sur différents serveurs, augmentant ainsi la segmentation et donc la résilience et la sécurité (pas les même machines en fonction du niveau de sensibilité de la donnée)
- Transparent pour les utilisateurs

Enfin, il est possible de coupler ce principe avec de la réplication synchrone (DFSR). Ainsi, l'utilisateur accède de façon transparente à un emplacement réseau permettant de faire de la haute disponibilité et de la répartition de charge.

II. Notions importantes

Pour bien cerner les principes clés du DFS et de la réplication associée, il est essentiel de comprendre les trois notions sur lesquelles il va se reposer :

- **Racine DFS** : C'est le point d'entrée principal du système DFS. Cette racine est un emplacement réseau qui contiendra les 'raccourcis' vers les cibles DFS gérée par le serveur. Si le serveur est intégré dans un domaine, cela ressemblera à '***\\domaine.ext\racineDFS***' et servira de point d'entrée sur les autres partages.

- **Dossier** : C'est le nom du partage affiché côté client et dans la configuration du serveur. Une liaison sera ensuite faite entre ce dossier et une cible afin de faire le lien entre ces deux éléments. Certains dossiers n'utilisent pas de cible mais sont uniquement présents pour structurer l'arborescence de la racine DFS. Ces dossiers sont également appelés '***Liaisons DFS***'.

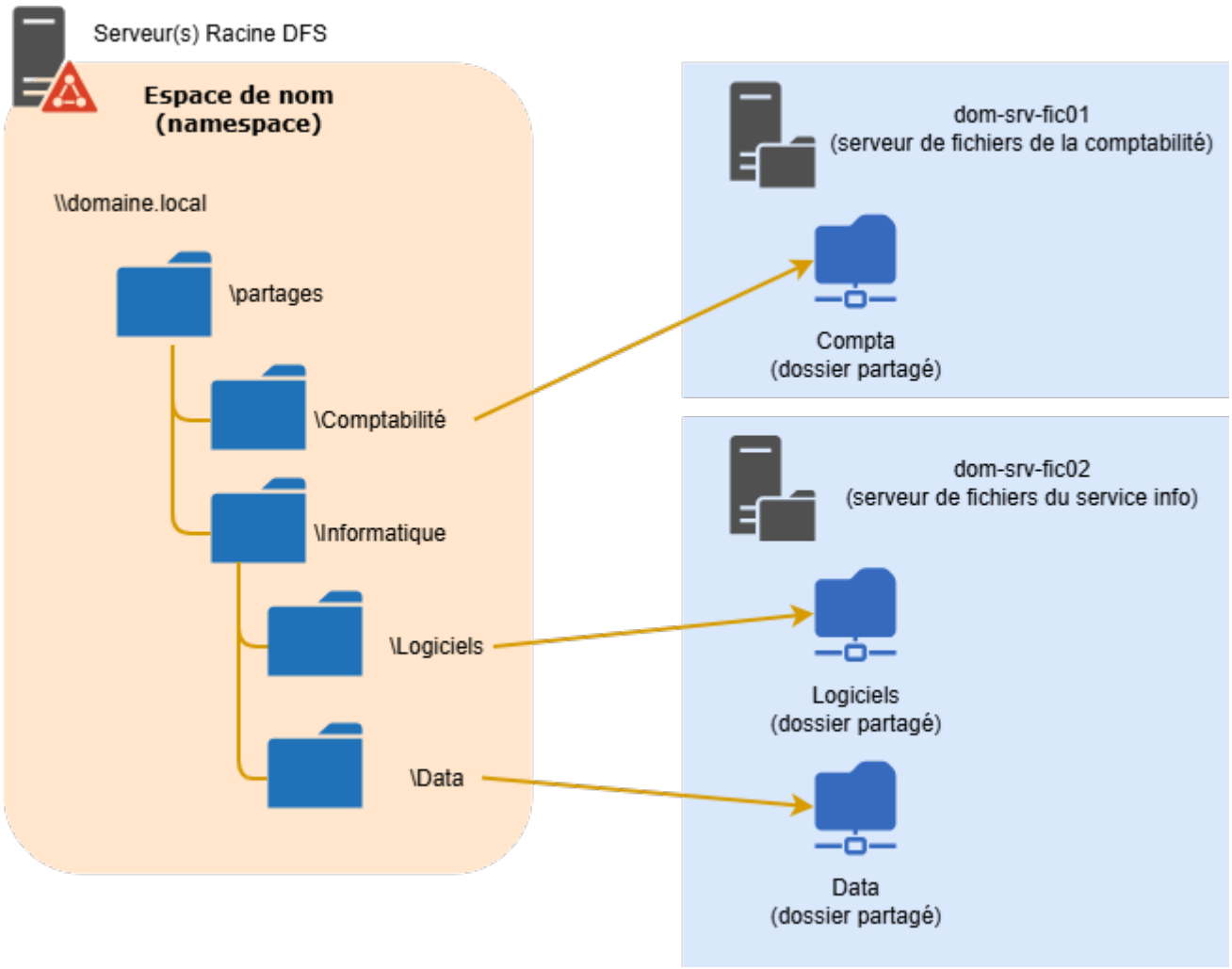
- **Cible** : Le serveur et le chemin réseau réel pour accéder au dossier partagé. Ce qui permettra lors de l'accès au dossier, de renvoyer l'utilisateur au bon endroit.

III. Architecture

Dans le domaine '***domaine.local***', il y a 3 partages qui doivent êtres accessibles aux utilisateurs.

- Comptabilité
- Informatique
  - Data
  - Sources

Voici comment se présente cette architecture :



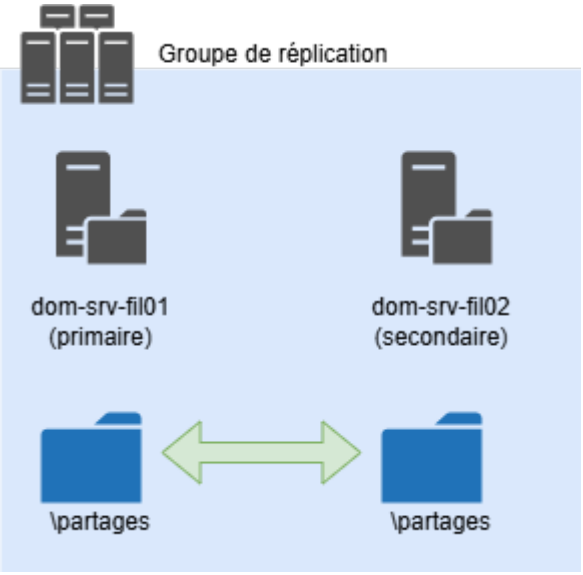
Ainsi du point de vue utilisateur, pour accéder aux logiciels, le chemin sera : '***\\domaine.local\partages\informatique\logiciels***'.

IV. DFSR

La réplication DFS ou ***DFSR*** est un système de réplication des fichiers s'appuyant sur DFS et utilisé en complément de celui-ci pour mettre en place de la haute disponibilité sur les partages de manière transparente pour les utilisateurs.

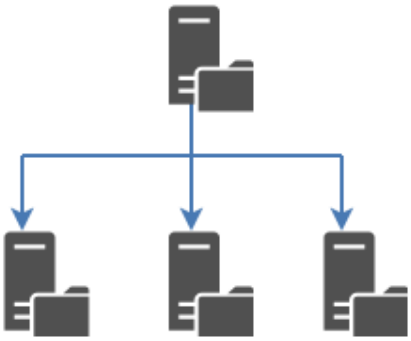
Un groupe de réplication est créé. Celui-ci contient les membres qui sont les serveurs partenaires de réplication.

Dans ce groupe, les dossiers partagés sont publiés avec des sources et des destinations. La réplication pourra se faire alors au choix en sens unique ou de façon bilatérale.



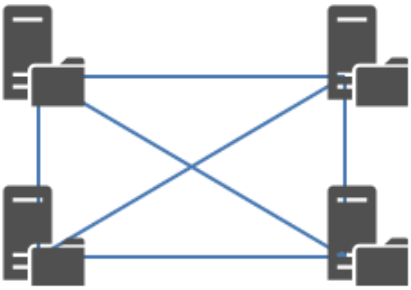
Il existe deux types de topologie de réplication :

Hub and spoke



Dans ce mode, un serveur central réplique de manière unilatérale vers les autres membres. Cela permet essentiellement de garder des copies des fichiers.
(nécessite 3 nœuds minimum)

Full Mesh



Dans ce mode, chaque serveur réplique de manière bilatérale avec ses autres partenaires. Cela permet d'assurer de la haute disponibilité.

Il sera ensuite également possible de décider si les réplications doivent se faire de façon synchrone ou asynchrone (créneaux définis dans un planning).

V. Conclusion

En conclusion, de par son architecture même, le système de racine DFS permet une gestion simple et centralisée de l'accès aux données. Lorsqu'il est en plus couplé à la mise en cache et à la réplication, cela permet aussi :

- **Simplification de l'administration** : Lorsqu'une cible DFS est arrêtée (panne, maintenance, remplacement de serveur), elle peut être facilement déplacée sur un autre serveur sans que cela ne change quoi que ce soit d'un point de vue utilisateur.

- **Intégration client** : La partie client est intégrée à windows (mise en cache, copie hors ligne), ce qui évite des installations supplémentaires.

- **Confort utilisateur** : Les données sont facilement accessibles, toujours au même endroit, de façon transparente. La mise en cache et la gestion des fichiers hors ligne permet la mobilité des utilisateurs, couplé à la technique de redirection des profils utilisateurs, cela garantit également à l'utilisateur qu'il accédera toujours à ses données et son environnement de travail, peu importe le terminal sur lequel il se connecte.

- **Sécurité :** Les permissions ne sont plus gérées par les permissions de partage, mais ce sont directement les permissions sur les dossiers / fichiers (**ACL**) qui s'appliquent.

- **Performance / Haute disponibilité** : Couplé à DFSR et à la mise en cache, cela permet une amélioration des performances ainsi qu'une disponibilité accrue des données. En effet, les données étant présente et synchronisées sur plusieurs serveurs, il est possible de répartir la charge entre ceux-ci et de garantir une bonne tolérance à la panne.