UNIVERSITÉ PERPIGNAN VIA DOMITIA

La Région Occitanie Pyrénées - Méditerranée

Solutions de haute disponibilité

Rapport de projet tuteuré

Licence Professionnelle AdmiSys

Année 2020-2021

Auteur : Nathalie Bomfleur

Tuteur : François Figarola

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidée lors de la mise en œuvre du projet et de la rédaction du présent rapport.

Tout d'abord, je remercie mon tuteur, M. François Figarola, qui m'a aiguillée et encouragée non sans humour.

Je remercie également mon compagnon pour ses conseils et son soutien.

Merci aussi à Mme El Yacoubi et Mme Calvet pour leur compréhension durant cette situation sanitaire exceptionnelle.

Table des matières

Introduction **4**
Solutions de haute disponibilité **5**
Network Load Balancing 6
Failover 7
Mise en Miroir 8
Transfert de journaux de transactions (Log Shipping) 9
Réplication 10
Always On 11
Infrastructure du projet « SQL Server Always On » **13**
Installations des logiciels et configuration préliminaires **14**
Installation du rôle Failover Microsoft Windows Server 14
Failover Manager 15
Active directory 19
Installation de SQL Server 2019 ..... **20**
Installation du SQL Server Management Service 25
Création d'un partage pour le groupe Always On 26
Création et configuration du groupe Always On **28**
Test de basculement **34**
Simulation de panne 35
Problèmes rencontrés lors de la mise en pratique **37**
Conclusion **39**
Sources **40**

Introduction

De nos jours, les bases de données sont omniprésentes dans les infrastructures informatiques professionnelles. Si elles sont parfois considérées accessoires, elles sont en fait indispensables si ce n'est vitales à l'activité de nombreuses entreprises. Une indisponibilité ou perte de données représenterait, dans ce cas-là, une catastrophe pour l'entreprise concernée. Il est donc évident que les technologies de haute disponibilité (high availability) et de reprise après désastre (Disaster Recovery) font partie intégrante de tout système de gestion de bases de données de qualité.

Commençons par deux définitions basiques.

Haute disponibilité : Le but de la haute disponibilité est d'éviter que le système soit inaccessible en cas de panne quelconque, ou du moins de minimiser le temps d'inaccessibilité. Il s'agit d'assurer la continuité d'un système ou service indépendamment de l'intervention humaine. En s'orientant d'après le Service Level Agréement-entente sur la qualité de service entre le prestataire de services informatique et son client- il est possible de définir des standards de continuité de service, tels que dans cette liste non exhaustive :

Classe de disponibilité	Pourcentage de disponibilité	Temps d'inaccessibilité toléré
3	99,9%	43,80 min : mois
4	99,99%	4,38 min/mois
5	99,999%	0,44 Min/mois

Source : Caesar, Daniel, et al. *SQL Server 2019 für Administratoren Das Umfassende Handbuch*. Rheinwerk Computing, 2020.

Reprise après désastre : Contrairement à la haute disponibilité dont le but est préventif, la reprise après désastre ou Disaster Recovery qui entre dans le plan de reprise d'activité de l'entreprise, vise à réparer les dégâts causés par une panne. Tout comme pour la haute disponibilité, il est possible de définir des standards. Dans le cas du disaster recovery, les buts sont fixés sous forme de Recovery Time Objective (RTO) – le temps maximal tolérable d'inaccessibilité d'une ressource informatique – et de Recovery Point Objective (RPO) – le taux maximal tolérable de perte de données en cas de sinistre.

Le sujet de ce rapport portant sur Always On, un ensemble de technologies visant à améliorer la haute disponibilité, le sujet de Disaster Recovery ne sera pas plus approfondi par la suite.

Solutions de haute disponibilité

Il existe plusieurs moyens de mettre en place une politique de haute disponibilité sous Microsoft SQL Server.

- **Services de basculement cluster (Failover-Cluster)**
- **Mise en miroir de base de données**
- **Envoi de journaux de transactions**
- **Réplication**
- **Always On**

Les solutions de haute disponibilité sont parfois classées dans des catégories de « veille » (standby) :

Hot Stand-by : solution de synchronisation qui peut permettre une continuité de service par basculement automatique ou manuel

Warm Stand-by : solution de synchronisation qui nécessite une intervention « manuelle » pour assurer la continuité

Cold Stand-by : solution sans synchronisation pour laquelle le basculement ne peut pas être effectuée automatiquement. S'applique uniquement ou données en accès lecture seule qui ne sont jamais modifiées

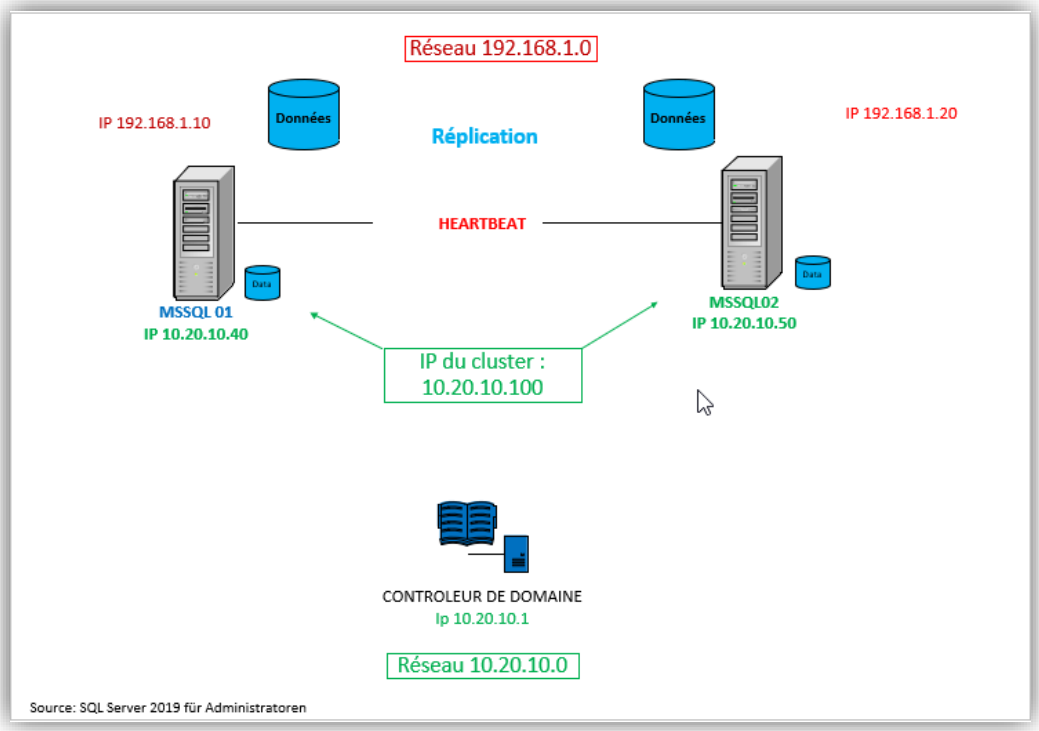
Les services de cluster fonctionnent sur le principe d'agrégation de plusieurs machines (« nœuds ») exécutant la ou les mêmes fonctionnalités avec les mêmes bases de données. Sur le réseau, ces machines se présentent comme un seul ordinateur avec un point d'accès (adresse IP virtuelle et,éventuellement, un nom DNS virtuel, Virtual Network Name) unique. Les ordinateurs clients s'adressent à ce point de contact pour accéder au service concerné. Le groupe d'ordinateurs dispose d'un stockage commun. L'état de santé des nœuds est surveillé et enregistré sur un volume de quorum Lors d'un fonctionnement normal, seul le nœud actif a accès aux données stockées. Dans les cas d'un incident, cet accès est basculé vers une ou plusieurs autres machines qui prennent ainsi le relais et assurent la continuité du service. Les configurations qui déterminent à partir de quand le basculement doit être effectué se trouvent également sur le volume de quorum. Le basculement est transparent pour les clients ; lors d'une connexion grâce à l'IP virtuelle ou au nom DNS, personne ne se rend compte qu'un ordinateur autre que le nœud primaire fournit le service.

Le clustering fait partie des rôles et fonctionnalités de Windows Server. Dans le cas d'un cluster de server SQL, SQL server utilise les fonctionnalités de Windows Server pour assurer la disponibilité de l'instance SQL. L'instance entière représente donc la ressource basculée dans le cluster.

Il existe deux types de services de cluster sous Windows : le cluster Network Load Balancing et le cluster Failover. Les deux ont un but principal différent.

Network Load Balancing

Le service cluster NLB vise à réduire la charge du réseau en distribuant les requêtes des clients sur plusieurs machines.



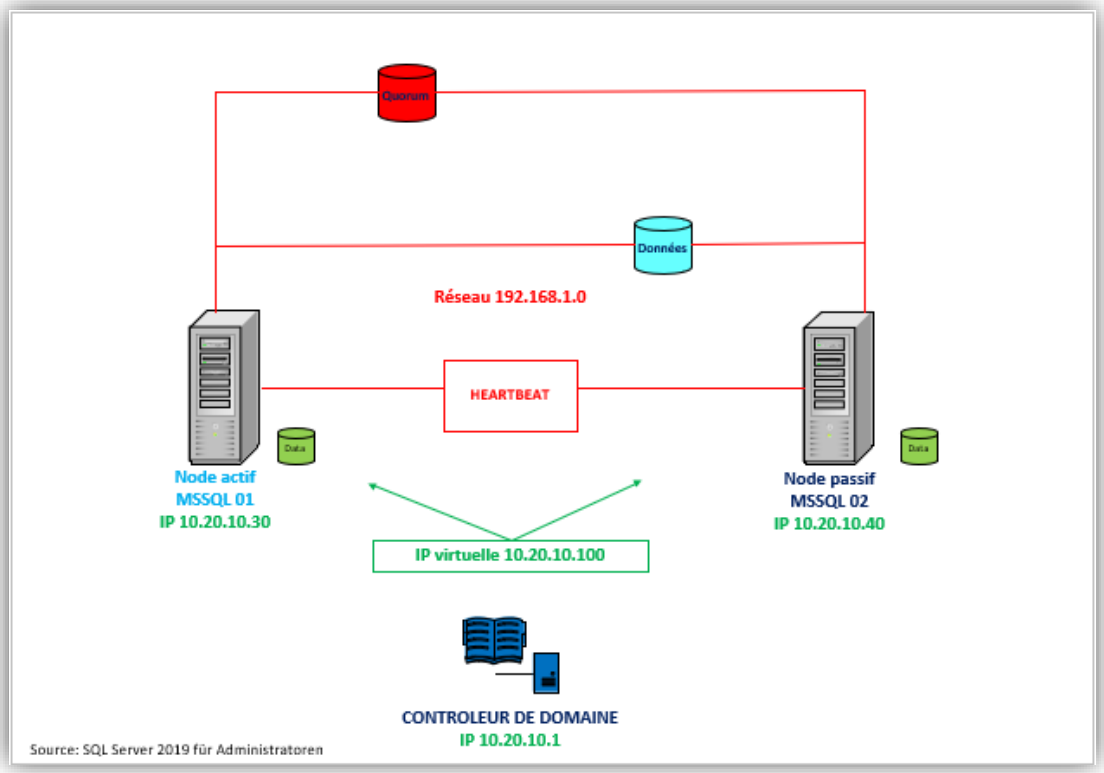
Dans cet exemple, la base de données est présente sur deux instances de SQL Server indépendantes. Il existe deux réseaux, 192.168.1.0 et 10.20.10.0. Le réseau 192.168.1.0 permet aux serveurs de communiquer entre eux pour mettre à jour la base de données. Les deux serveurs sont « joignables » sous l'adresse IP virtuelle 10.20.10.100 à laquelle sont adressées les requêtes client. Le service cluster détermine lequel des deux serveurs traite la requête. Les transactions sont répliquées pour que les deux bases de données soient à jour.

Le cluster NLB concerne une instance entière. Cette solution nécessite beaucoup de matériel et d'espace de stockage pour la réplication des bases de données concernées.

Ce type de cluster ne sera pas utilisé lors du présent projet.

Failover

Le cluster failover ne vise pas à réduire la charge du réseau mais à assurer la continuité du service SQL en cas de panne sur le serveur principal. Comme pour le cluster NLB, deux instances de SQL Server répondent sur une adresse virtuelle commune.



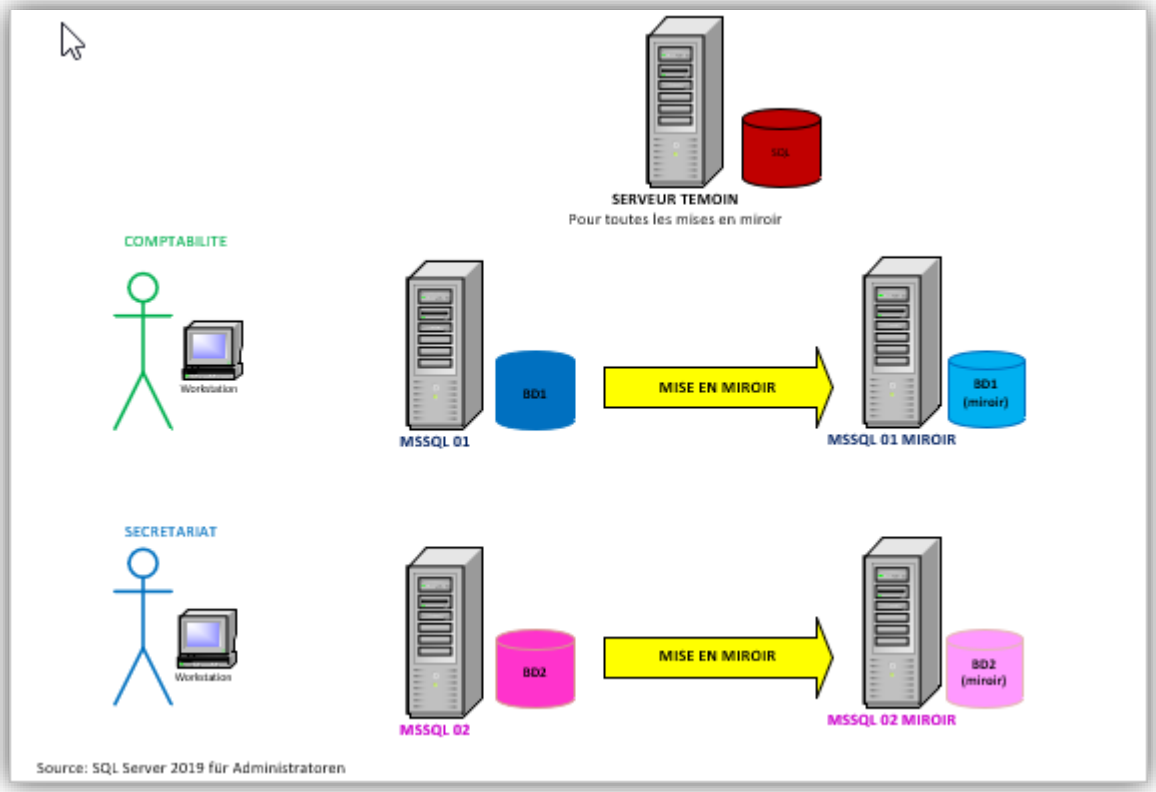
Contrairement au cluster NLB, il n'existe qu'un exemplaire de la base de données qui se trouve sur un espace de stockage partagé SAN. C'est le nœud actif qui a les droits d'accès exclusifs sur l'emplacement. Le stockage de la base de données peut s'effectuer à base de technologies ISCSI ou Fibre Channel. Le « heartbeat » interroge les deux instances pour déterminer si elles sont opérationnelles. Si un des serveurs ne répond pas au « heartbeat » en raison de panne, le processus de Failover est initié et l'autre serveur prend le relais.

Le Quorum fait partie du cluster manager et permet d'assurer l'intégrité des données en enregistrant en permanence l'état du cluster entier. Il se trouve également sur un espace de stockage partagé. Un cluster peut disposer d'un seul quorum ou d'un quorum par serveur.

Le cluster failover est une solution coûteuse mais complète qui permet de sécuriser une instance entière. Cependant, le volume représente un point de défaillance unique (Single Point Of Failure) qui rend cette solution « fragile » ;

Mise en Miroir

A l'inverse du Cluster Failover, la mise en miroir ne s'applique pas à une instance entière mais à une ou plusieurs bases de données. Les bases de données systèmes sont exclues.



Lors de la mise en miroir, les serveurs remplissent les « rôles » suivants : L'instance principale contient la base de données originale. Celle-ci est mise en miroir, donc répliquée en mode récupération, sur l'instance miroir. Un serveur témoin peut assurer la surveillance, similairement au quorum dans le scénario du Failover cluster. Cette machine vérifie la disponibilité des deux autres instances grâce au « heartbeat ». En cas d'indisponibilité de l'une des deux, elle déclenche la bascule failover. La mise en miroir peut être effectuée sans serveur témoin, mais dans ce cas-là, la bascule ne sera pas automatique.

Il existe trois modes de mise en miroir : lors du mode **asynchrone « high-performance »**, un COMMIT est effectué après la transaction sur le principal, puis les modifications sont envoyées au serveur miroir. Lors du mode **synchrone**, le COMMIT est effectué sur le principal et le témoin après la transaction sur le principal. Le mode **synchrone « high-safety »** fonctionne comme le mode synchrone, mais nécessite un témoin qui déclenche la bascule automatique lorsque le principal est indisponible.

La mise en miroir est une solution Hot Standby qui ne nécessite pas de matériel supplémentaire pour rediriger les requêtes client en cas de panne. Ceci la rend moins coûteuse que la solution de Failover Cluster. Cependant, elle ne permet pas de

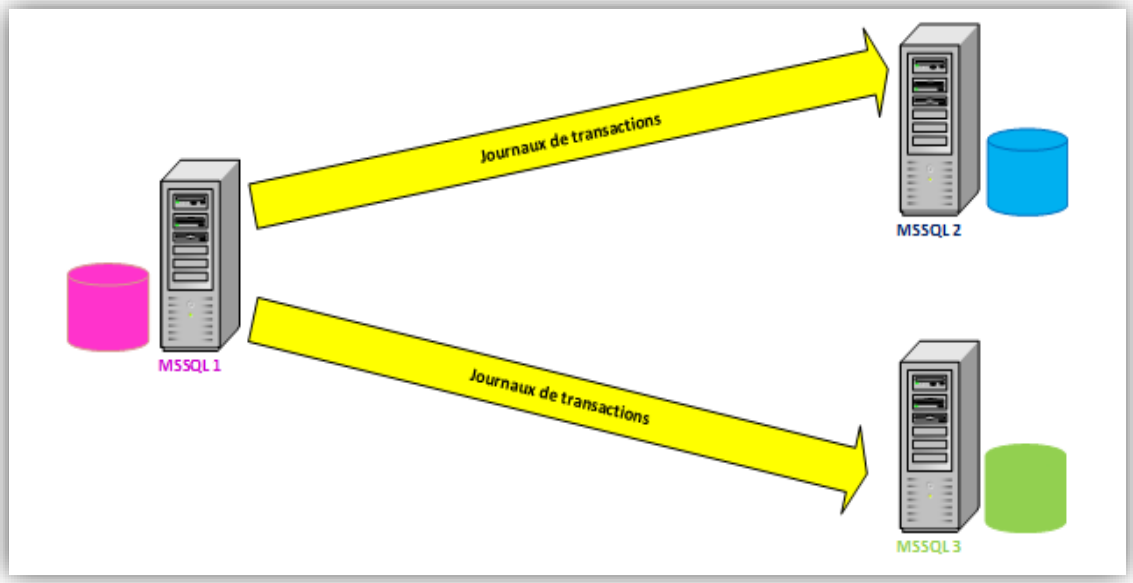
sécuriser des instances entières. Elle ne peut pas s'appliquer à de multiples bases de données, ni utiliser plusieurs serveurs secondaires.

Microsoft a annoncé que cette fonctionnalité ne sera plus proposée dans l'avenir.

Transfert de journaux de transactions (Log Shipping)

Tout comme la mise en miroir, le transfert de journaux de transactions concerne l'accessibilité au niveau des bases de données.

Les journaux de transactions sont régulièrement enregistrés sur le serveur principal, puis copiés sur les machines secondaires pour y être restitués par la suite. Ces fonctionnalités sont assurées par des « jobs » de l'agent SQL qui sont créés lors de la configuration du service de Log Shipping.

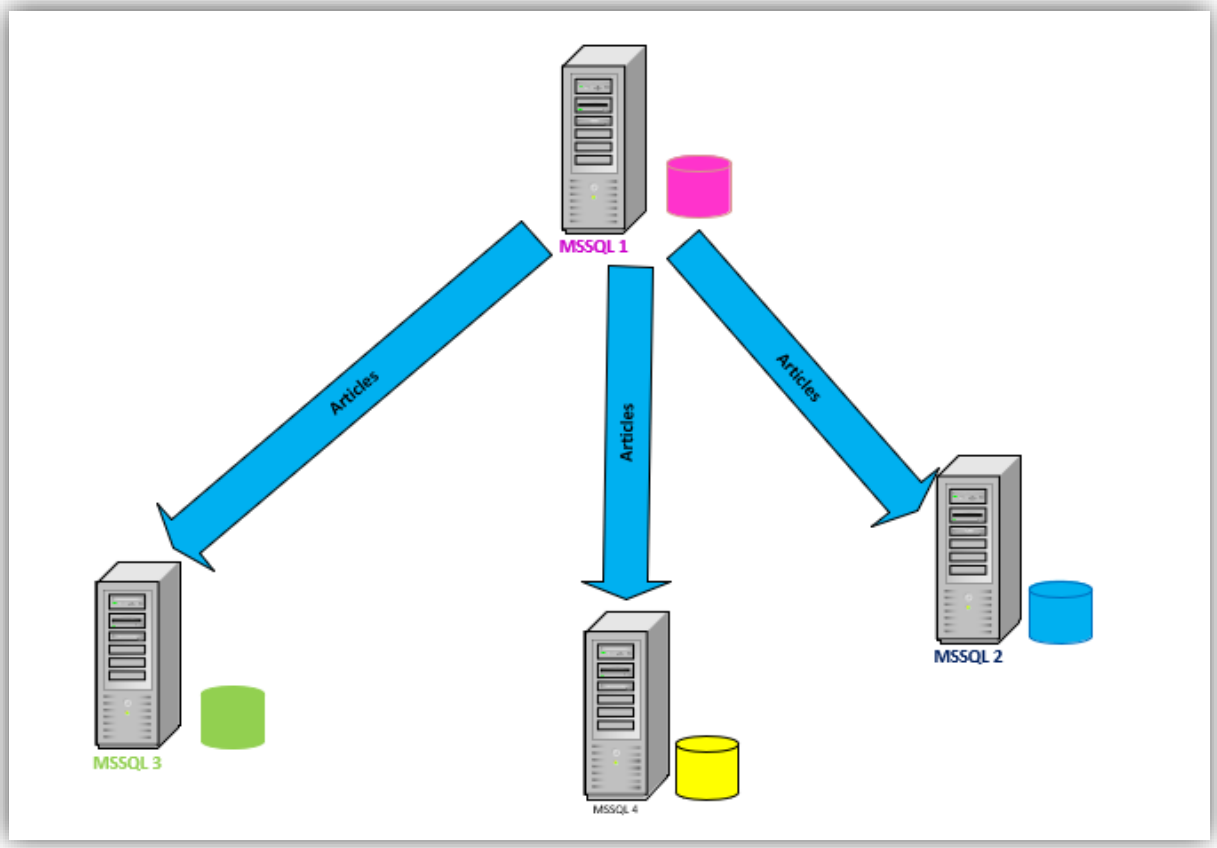


Le transfert de journaux de transactions peut être sujet à la perte de données et n'inclut pas de bascule lors de pannes.

Comme il n'existe pas de moyen de détecter des pannes, cette solution est une solution « warm-standby » qui nécessite l'intervention externe pour permettre un basculement.

Réplication

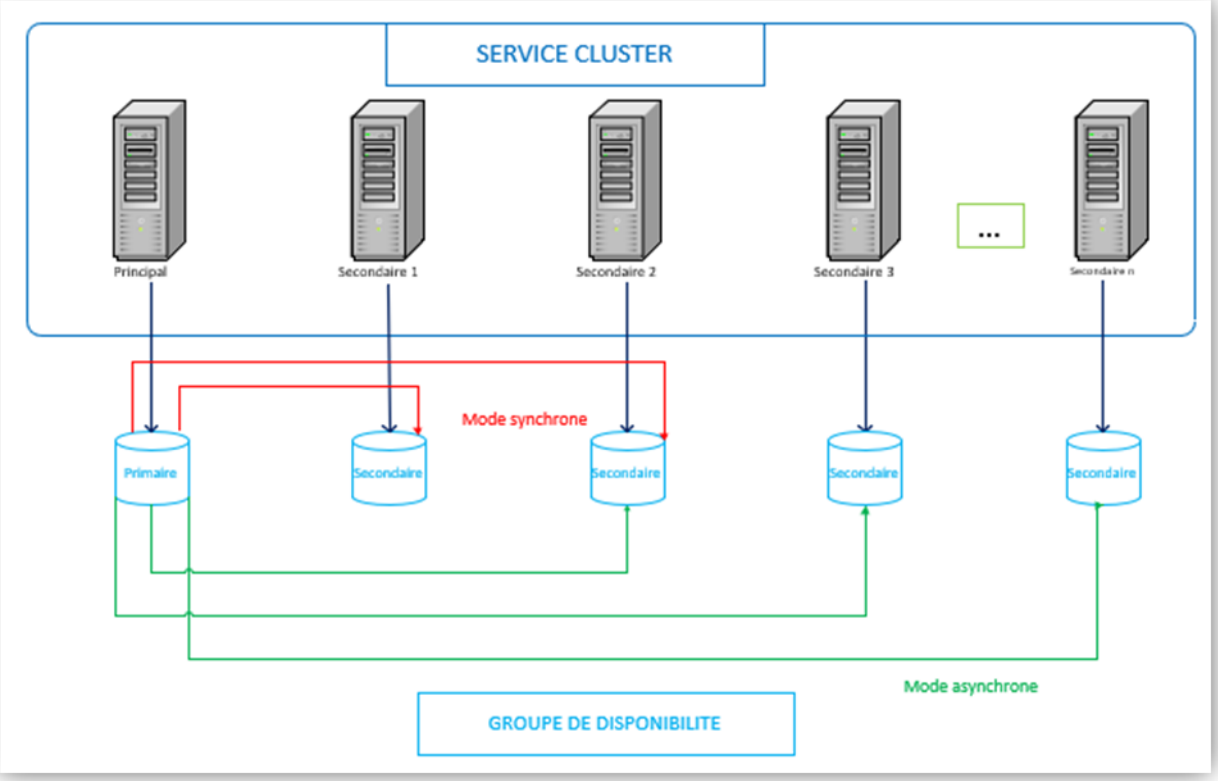
Lors de la réplication, un serveur principal agit comme un éditeur qui publie des données auxquelles sont abonnés un ou plusieurs serveurs secondaires. Ces données, appelés « articles », peuvent être des clichés de bases de données ou des transactions, et le moment de la réplication peut être configuré pour survenir à la suite de certains évènements déclencheurs (« trigger »).



Always On

Le terme « Always On » désigne un ensemble de fonctionnalités introduites avec SQL Server 2012 qui vise à combiner les avantages tout en remédiant aux défauts des technologies présentées préalablement. Il regroupe une version du Windows Failover Cluster améliorée et le concept des groupes de haute disponibilité (High Availability Groups). Par rapport aux éditions précédentes du service Failover Cluster, la version Always On permet (à partir de 2014) que tous les nœuds du cluster accèdent au volume partagé indépendamment les uns des autres. Le nombre de serveurs secondaires a également augmenté (4 en 2012, 8 en 2014), et il est possible d'étendre le basculement sur plusieurs sous-réseaux.

Les groupes de haute disponibilité constituent la suite logique de la mise en miroir. Ils permettent la haute disponibilité de plusieurs bases de données regroupées. Always On se base sur la technologie du cluster Failover mais ne nécessite pas d'espace de stockage partagé. Contrairement à la mise en miroir, Always On permet de sécuriser une instance entière. Comme pour la mise en miroir, les bases de données systèmes ne peuvent être rendues accessibles en haute disponibilité.



Le Cluster dispose également d'une adresse IP virtuelle. Un cluster Always On peut être composé de jusqu'à cinq nœuds. Un serveur est le nœud principal, les autres reçoivent les transactions du principal. Always On permet de spécifier des « groupes de disponibilité » dans lesquels on détermine quelles bases de données sont synchronisées dans quel mode. Les nœuds secondaires sont uniquement accessibles en lecture.

Les modes de synchronisation sont les même que pour la mise en miroir : lors du mode synchrone, la base de données primaire n'enregistre les transactions que lorsque les autres nœuds les ont reçues également. Lors du mode asynchrone, le nœud primaire exécute d'abord le commit puis transmets les transactions aux nœuds secondaires.

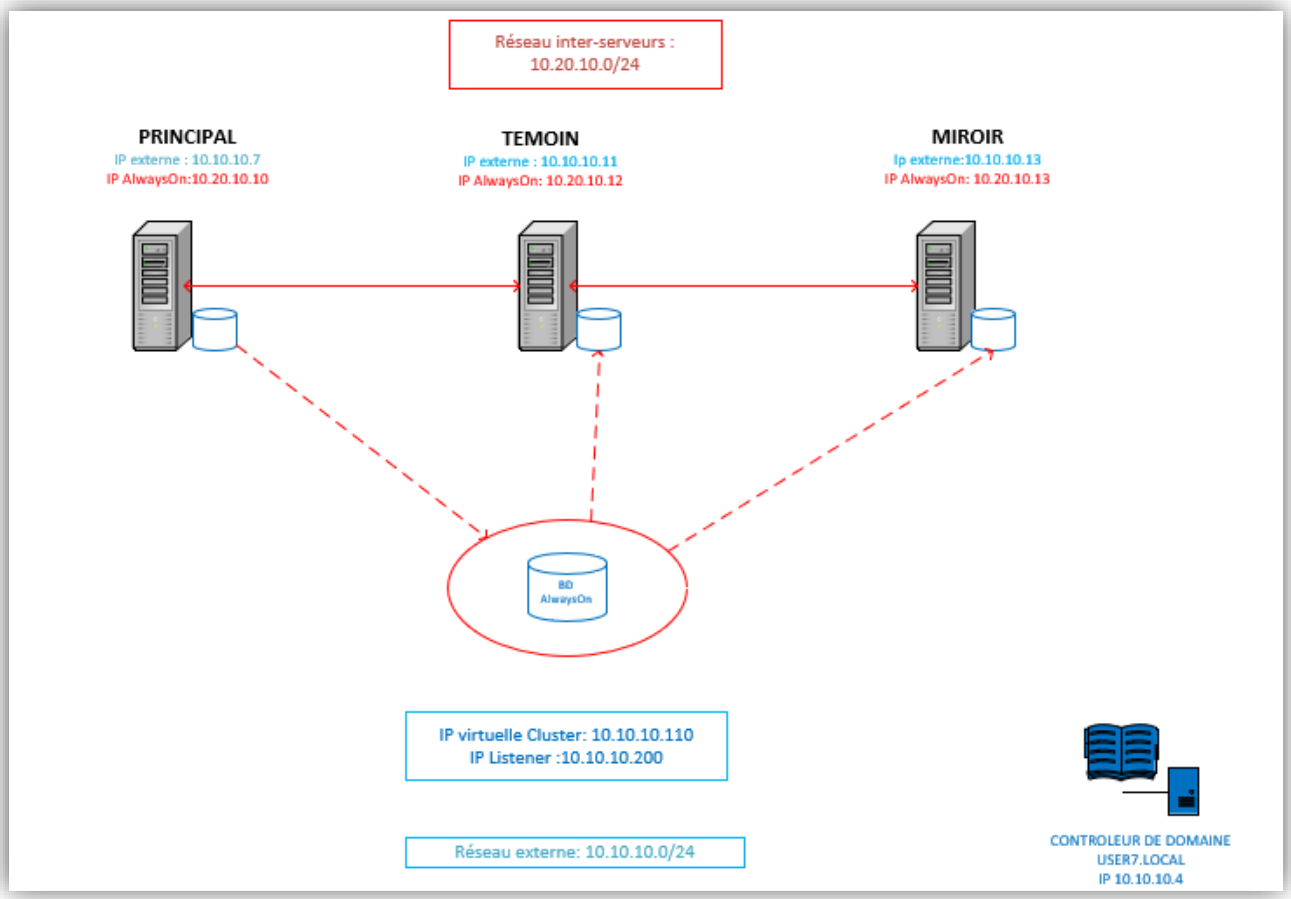
Ce présent rapport présentera une mise en pratique de groupes de haute disponibilité Always On.

Infrastructure du projet « SQL Server Always On »

Le scénario Always On a été créé dans un environnement virtuel sous VMware Esxi. Pour tester la fonctionnalité Always On, j'ai créé 4 machines virtuelles sous Windows Server 2019 :

- Un contrôleur de domaine DCUSER7
- Une instance primaire de SQL server PRINCIPAL
- Deux instances secondaires de SQL Server MIROIR et TEMOIN

Les serveurs PRINCIPAL, MIROIR et TEMOIN font partie du domaine USER7.LOCAL. Ils disposent chacun de deux cartes réseau. Les switchs virtuels sous VMware me permettent d'utiliser des réseaux séparés pour mes infrastructures. Les serveurs sont reliés à deux réseaux distincts : 10.10.10.0/ 24 pour le trafic « global », et 10.20.10.0/24 dédié exclusivement au trafic intra-serveurs (heartbeat, synchronisation).

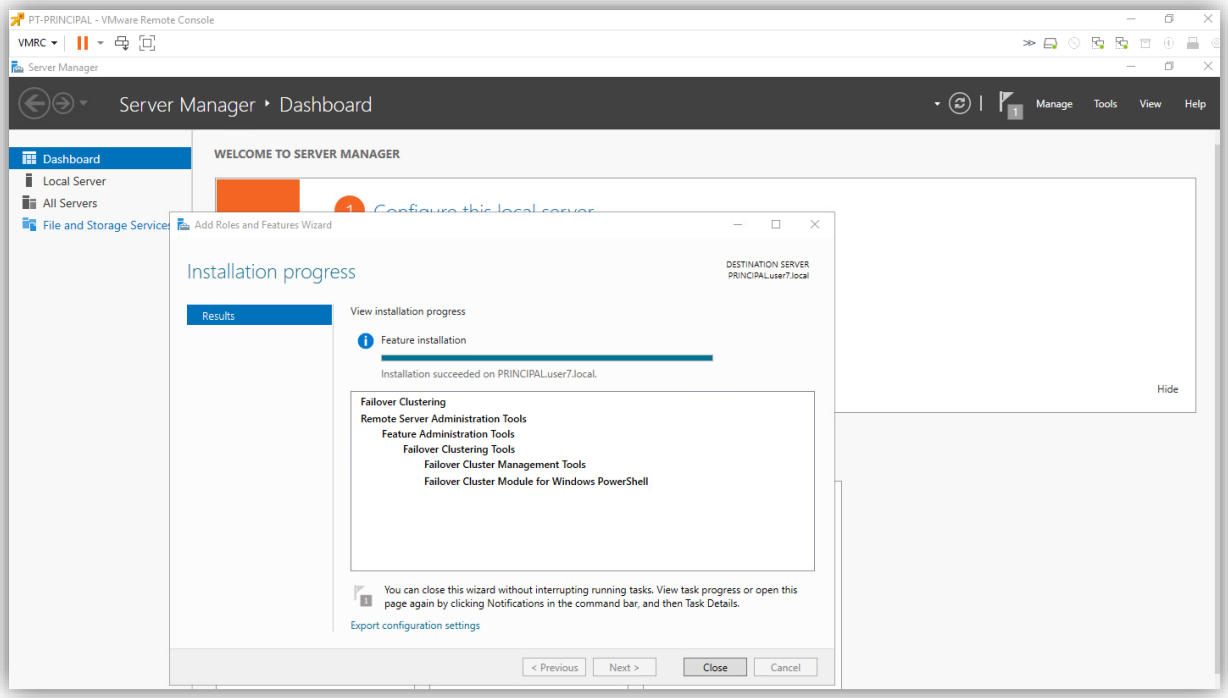


Installations des logiciels et configuration préliminaires

Les couches logicielles suivantes sont installées à l'identique sur les trois serveurs de l'infrastructure.

Installation du rôle Failover Microsoft Windows Server

Afin de pouvoir utiliser les services de haute disponibilité SQL Server, il est nécessaire d'avoir créé un Cluster sous Windows Server. L'installation de ce rôle peut s'effectuer par l'interface graphique

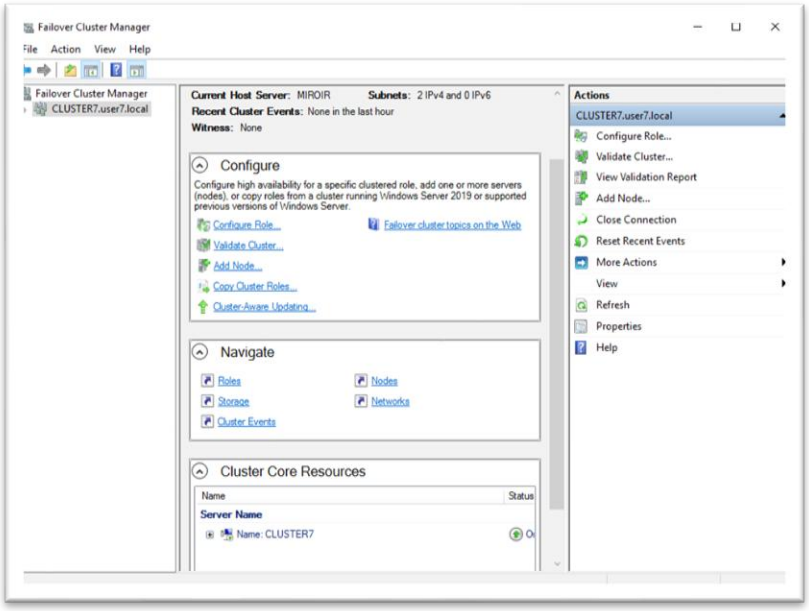


Ou en ligne de commande grâce à la commande PowerShell suivante :

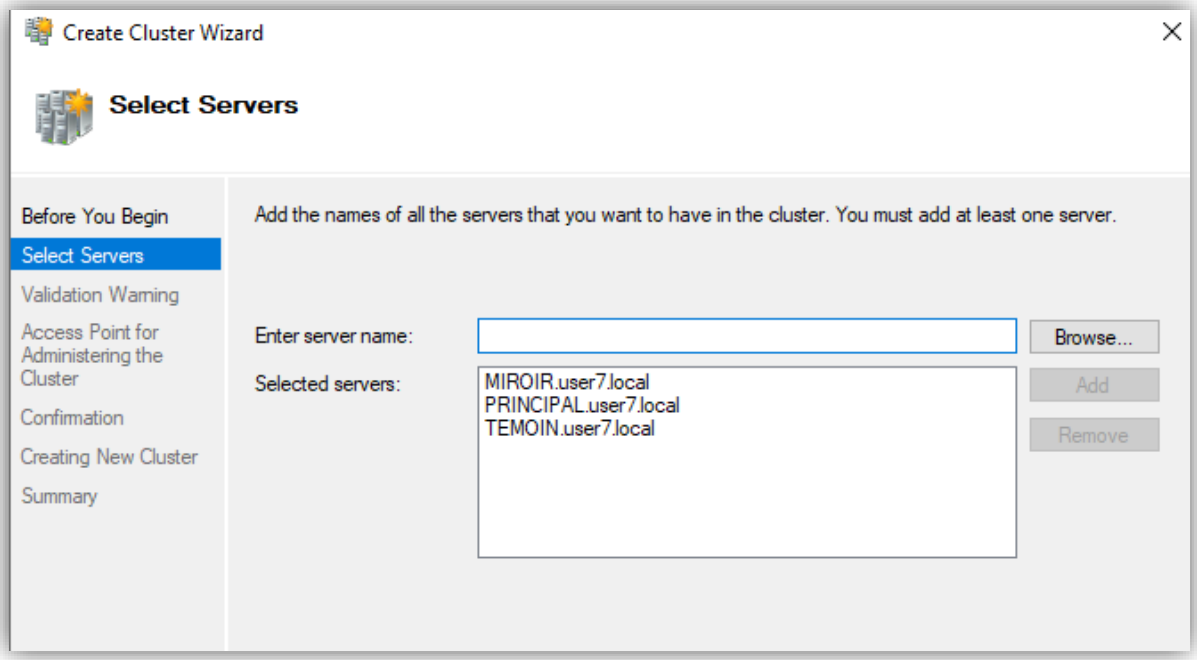
```
PS C:\Users\Administrator> Install-WindowsFeature -Name "Failover-Clustering" -IncludeAllSubFeature

Success Restart Needed Exit Code      Feature Result
------- -------------- ---------      --------------
True    Yes            SuccessRest... {Failover Clustering}
WARNING: You must restart this server to finish the installation process.


PS C:\Users\Administrator>
```

Failover Manager

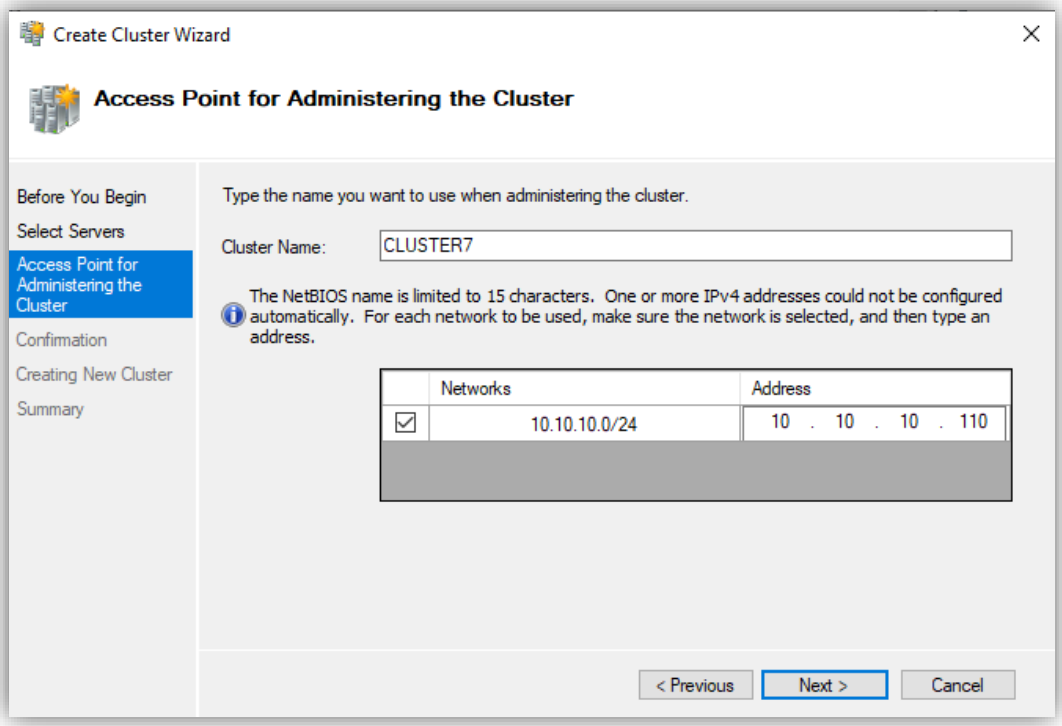
La configuration du Cluster s'effectue ensuite par la console "Failover Manager" :



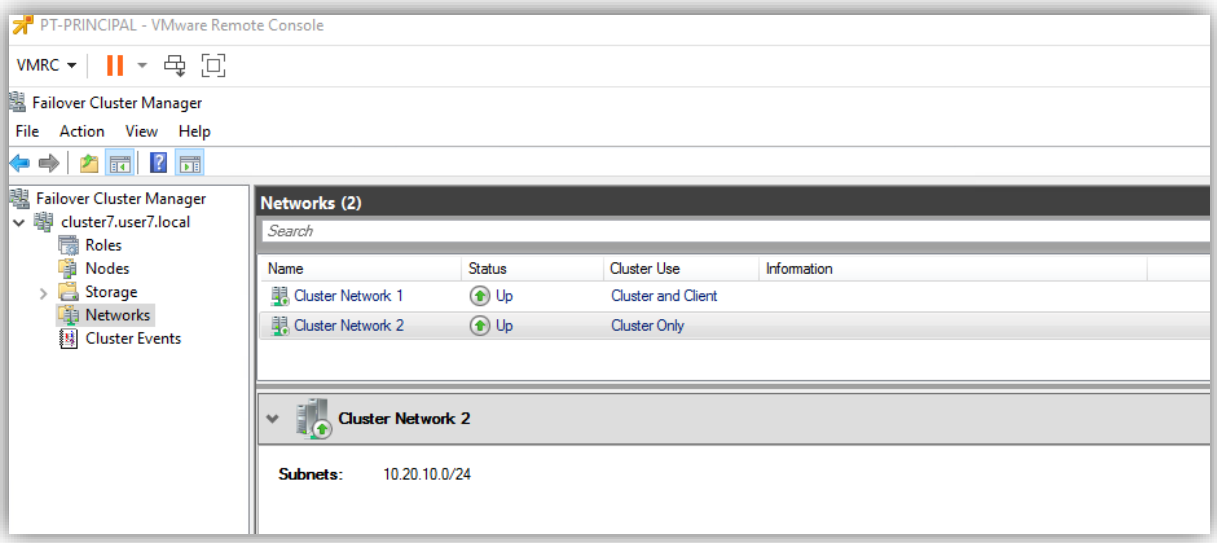
La première étape est d'ajouter les machines concernées



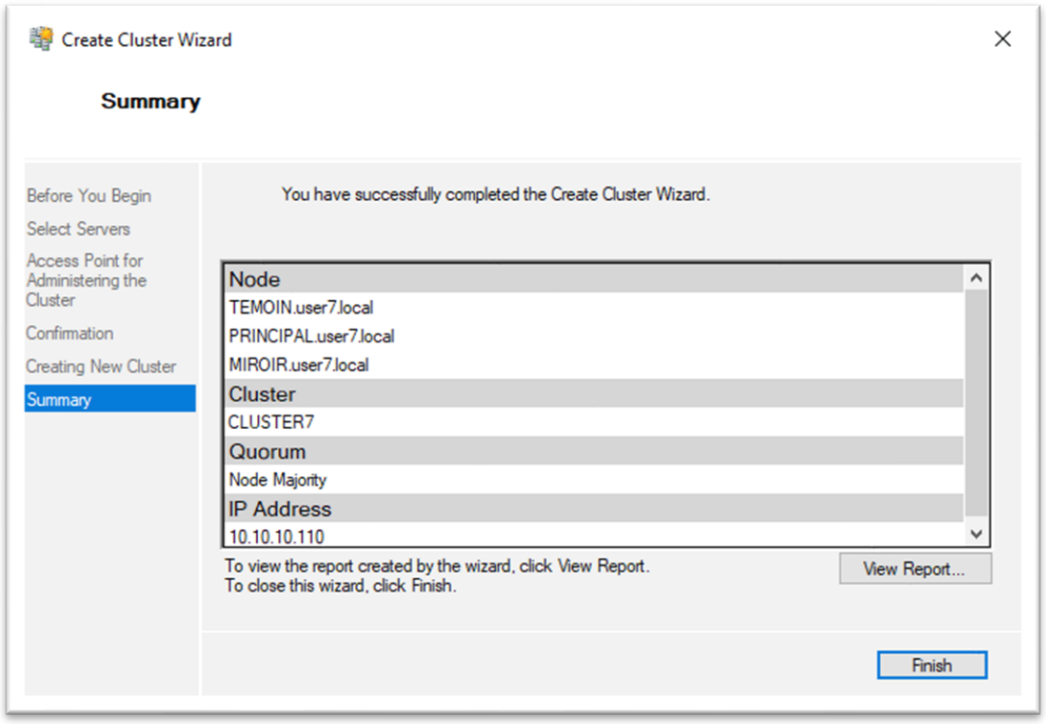
Par la suite, le nom du cluster et l'adresse virtuelle sont définis. C'est sous cette adresse IP que le cluster sera joignable par les applications clients.



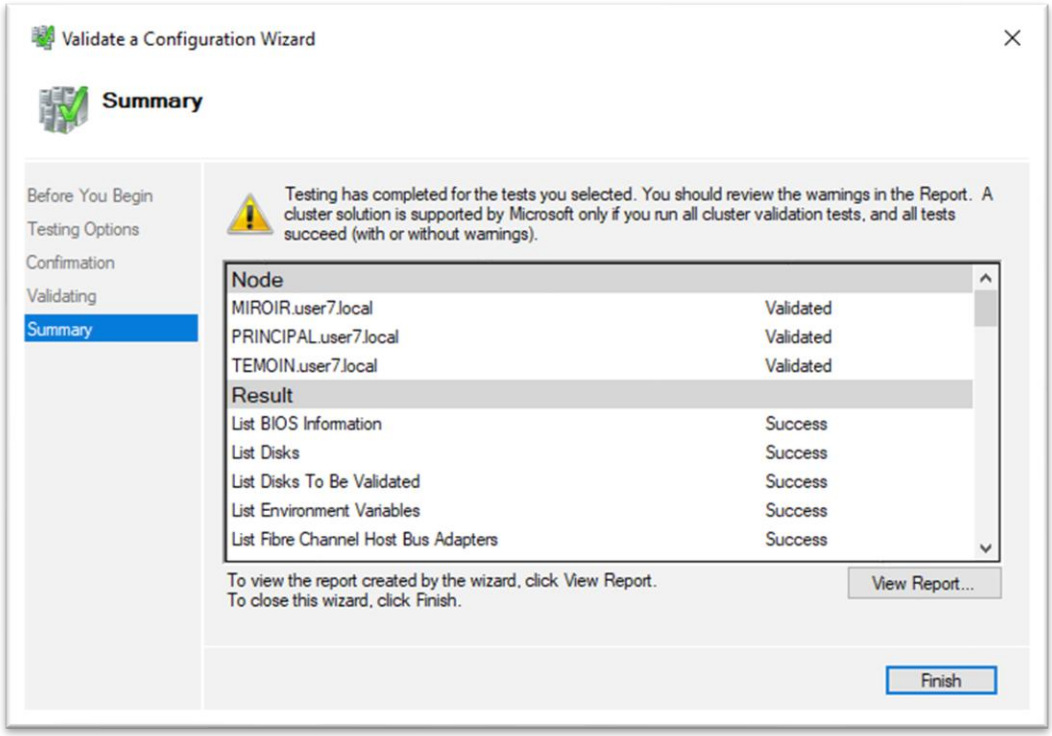
La console Failover Manager détecte le deuxième réseau dédié aux communications entre serveurs



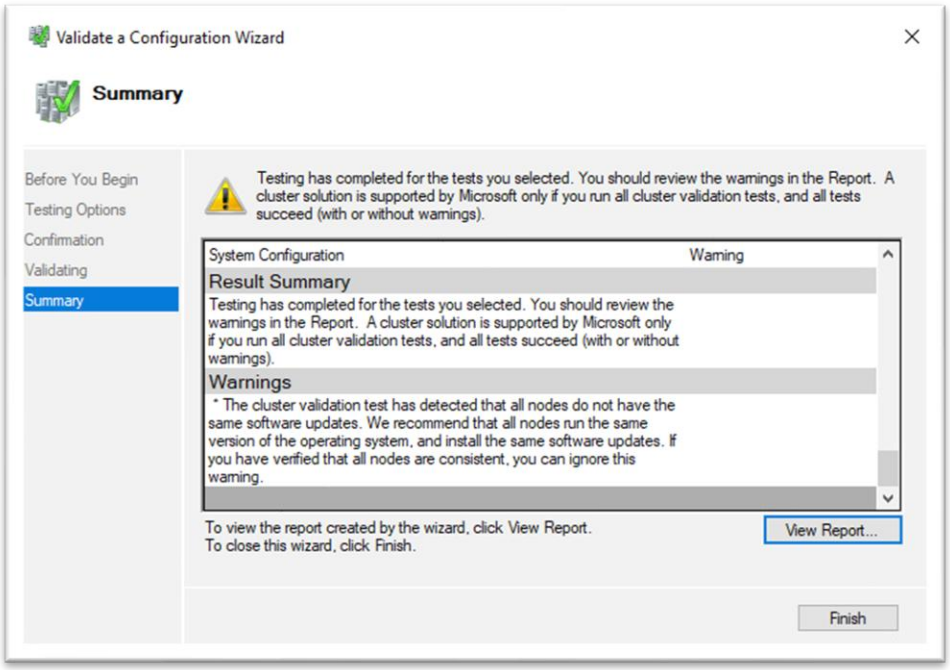
Le logiciel détecte automatiquement le mode de quorum adapté : comme l'infrastructure dispose d'un nombre impair de nœuds, le quorum décidera par « vote majoritaire » (majority node) à partir de quand le basculement sera effectué. Dès que deux serveurs détecteront que l'état de santé du serveur principal n'est pas suffisant, un d'entre eux prendra le relais.



Le programme de configuration vérifie les paramètres et affiche d'éventuels problèmes



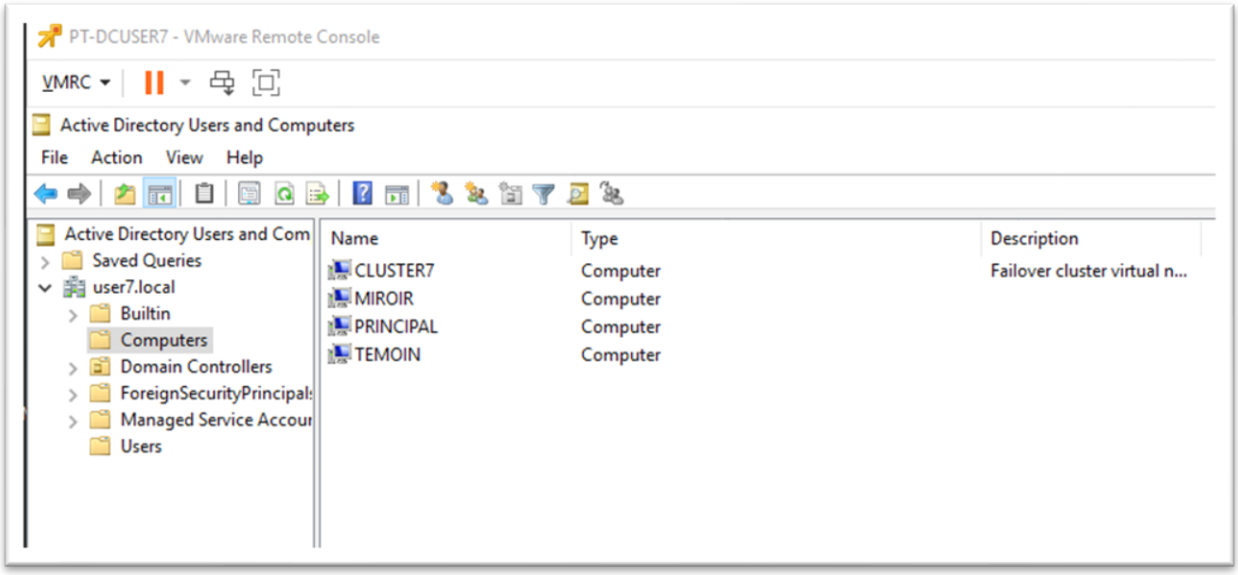
Ici, l'assistant affiche des avertissements. Après vérification, il s'agit de mises à jour non effectuées :



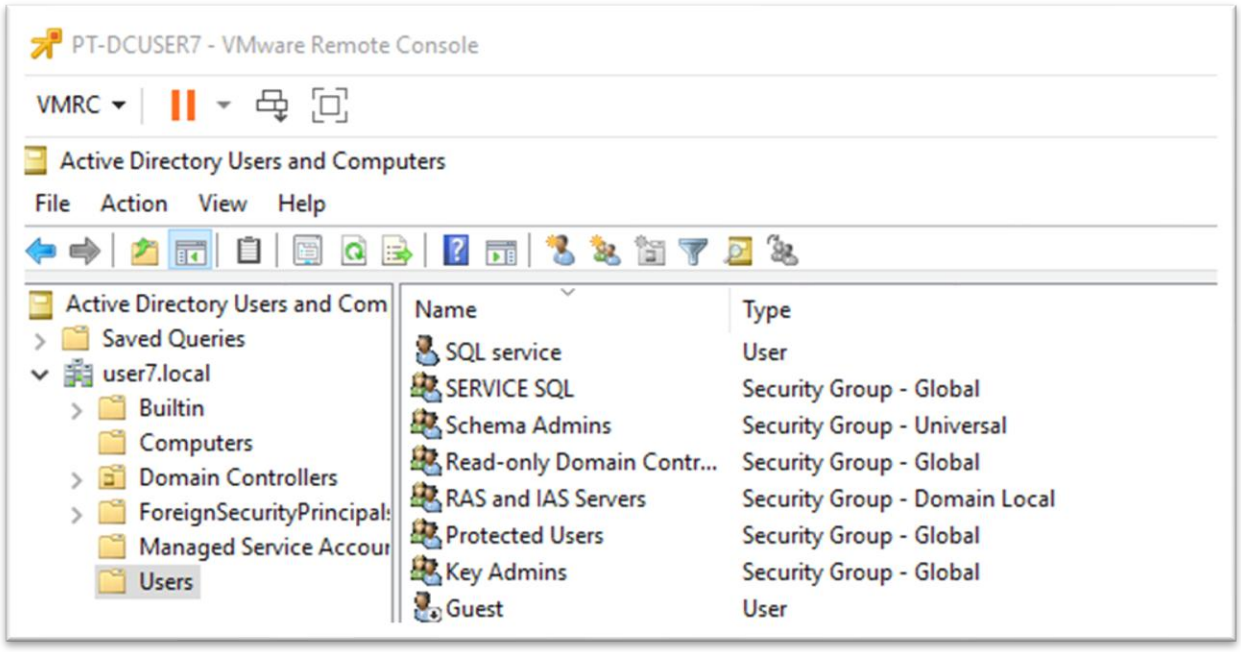
Ce problème n'empêche pas la création et le bon fonctionnement du cluster.

Active directory

Un compte Active Directory est créé automatiquement pour le cluster :

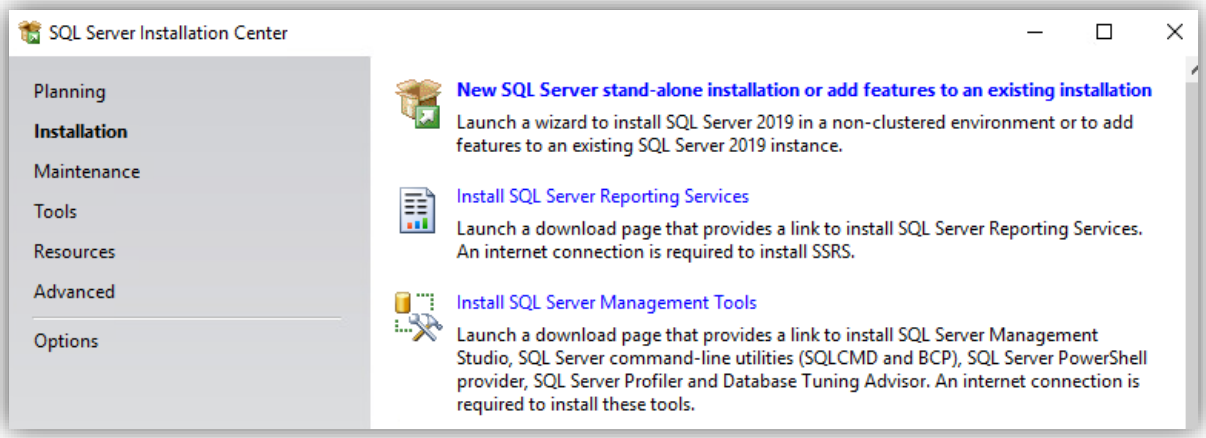


Afin de faciliter les connexions, j'utilise un compte Active Directory pour toutes les connexions sur SQL Server. Ceci ne pose pas de problème dans un environnement test, Cependant dans un environnement de production, il est préférable d'utiliser un compte par instance SQL. Il est également possible d'intégrer les comptes services dans un groupe AD et d'accorder les droits de connexion à ce dernier.



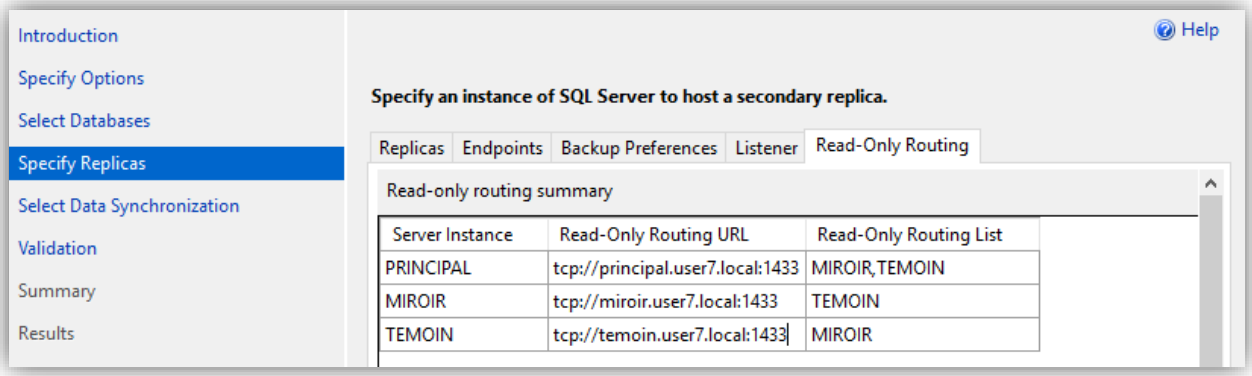
Installation de SQL Server 2019

J'effectue la même installation d'une seule instance sur les trois serveurs

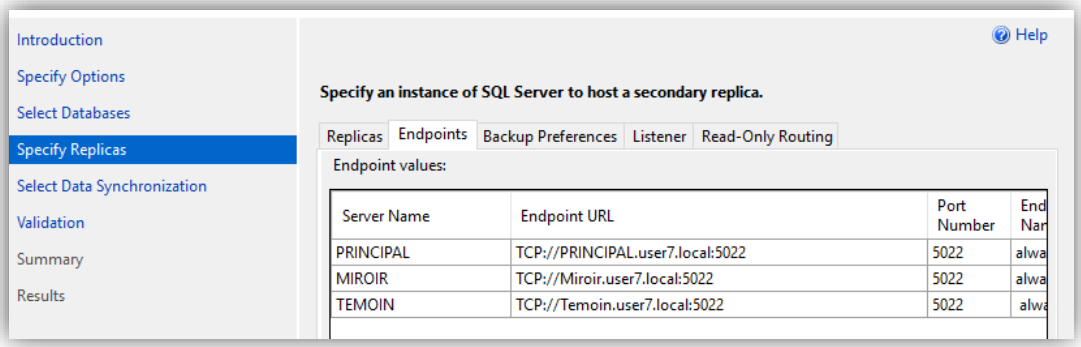


Un avertissement concernant le pare-feu s'affiche. Il est, en effet, primordial de bien configurer les ports ouverts au services SQL. Par défaut, SQL server utilise le port 1433 pour les communications en TCP et le port 5022 pour les connexions aux points de terminaison.

Port 1433 :

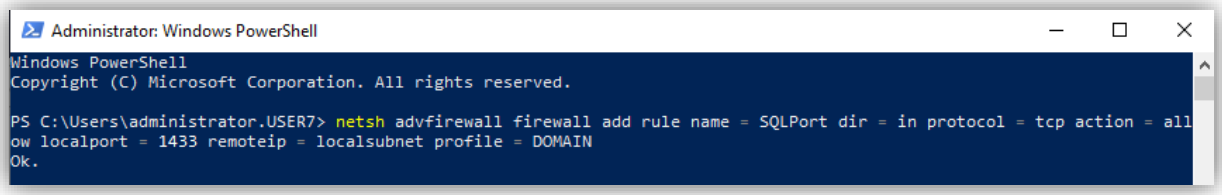


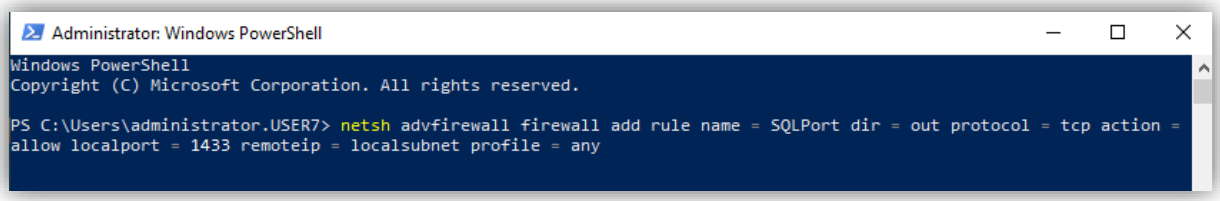
Port 5022 :



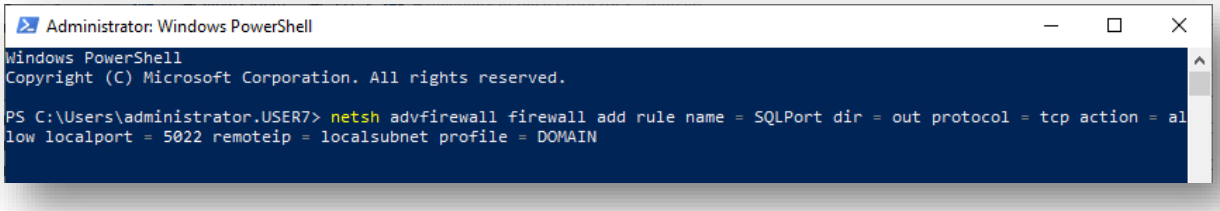
Je crée donc une règle de pare-feu pour ouvrir les ports nécessaires au trafic sortant et entrant. Les ports utilisés par défaut par SQL serveur sont les ports 1433 et 5022 en protocole TCP

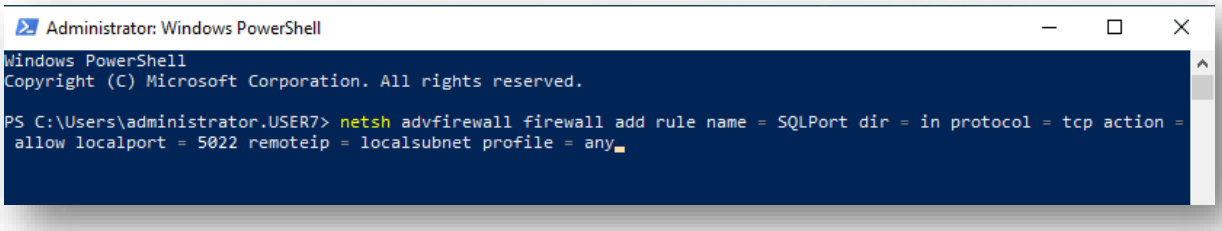
Port 1433



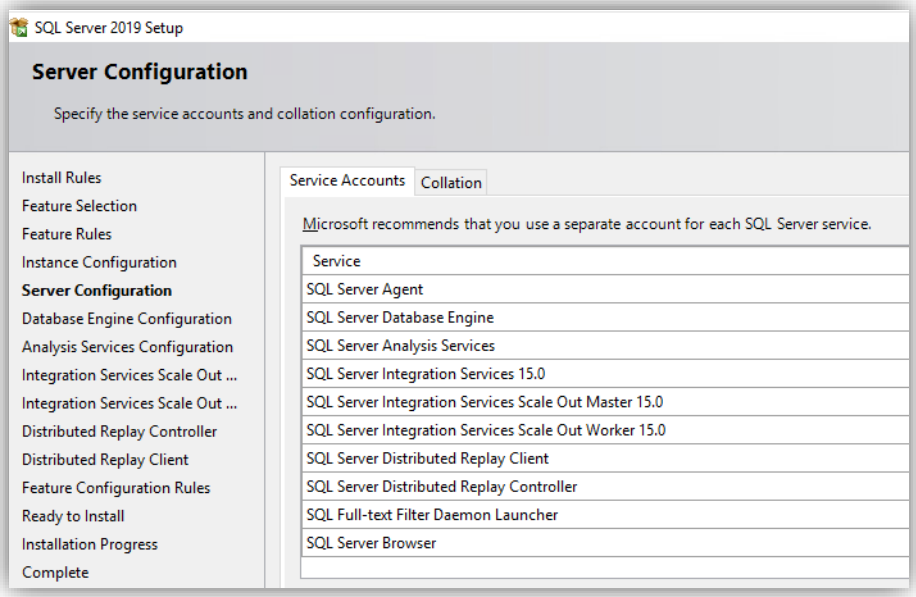


Port 5022

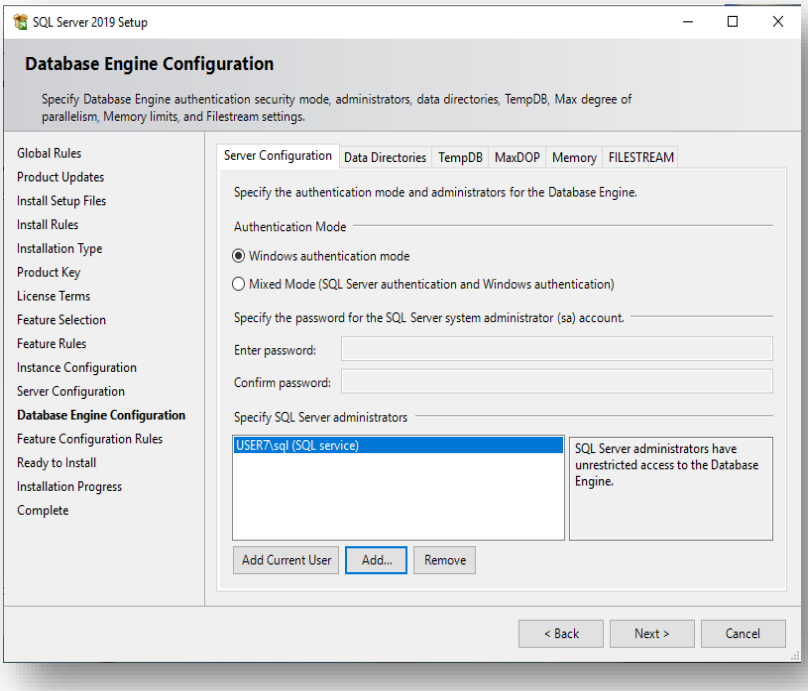

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\administrator.USER7> netsh advfirewall firewall add rule name = SQLPort dir = out protocol = tcp action = al
low localport = 5022 remoteip = localsubnet profile = DOMAIN

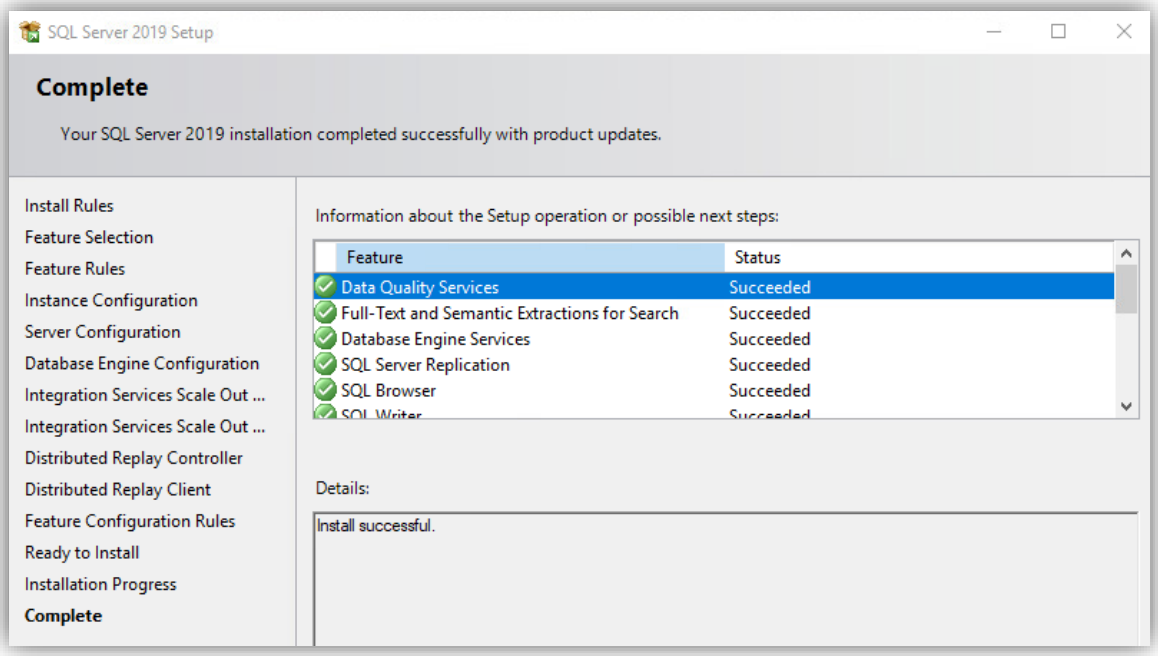

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\administrator.USER7> netsh advfirewall firewall add rule name = SQLPort dir = in protocol = tcp action =
allow localport = 5022 remoteip = localsubnet profile = any

L'installation de SQL server doit comprendre à minima le service Moteur de bases de données (Database Engine). Je choisis les services supplémentaires suivants :

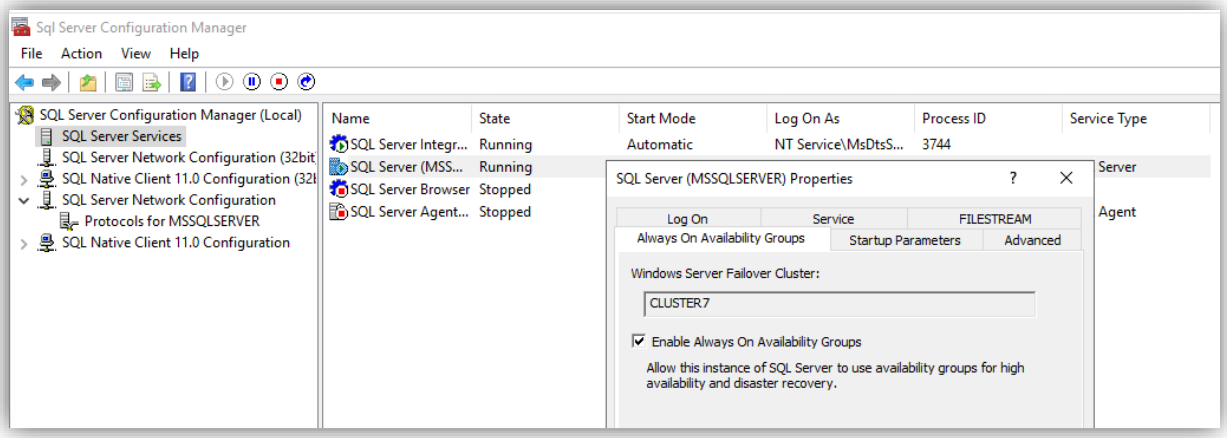


Je choisis l'authentification Windows et le compte AD SQL crée à cet effet.

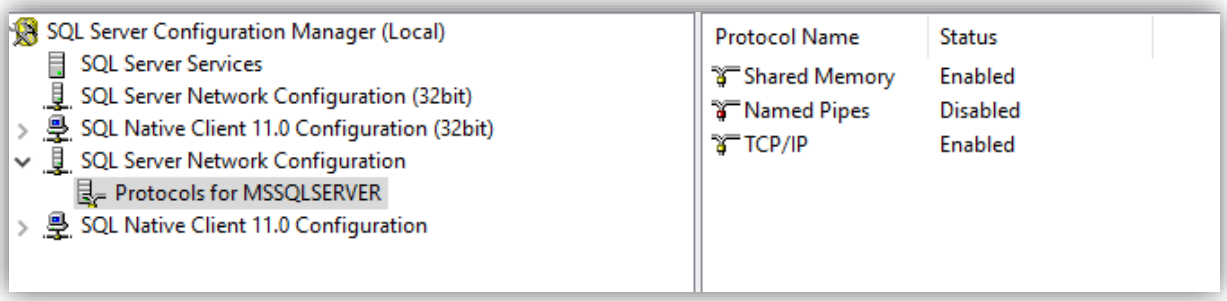




Pour le nouveau service SQL, j'active la fonctionnalité Always On

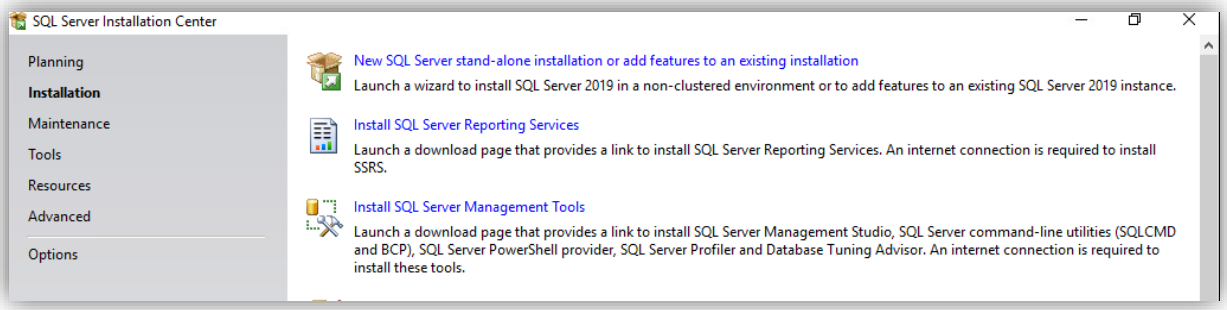


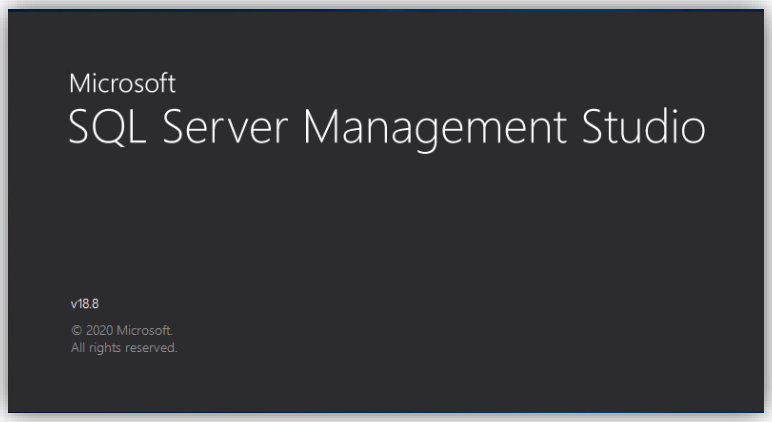
Et également les protocoles nécessaires à la communication entre serveurs (TCP/IP) et la mémoire partagée

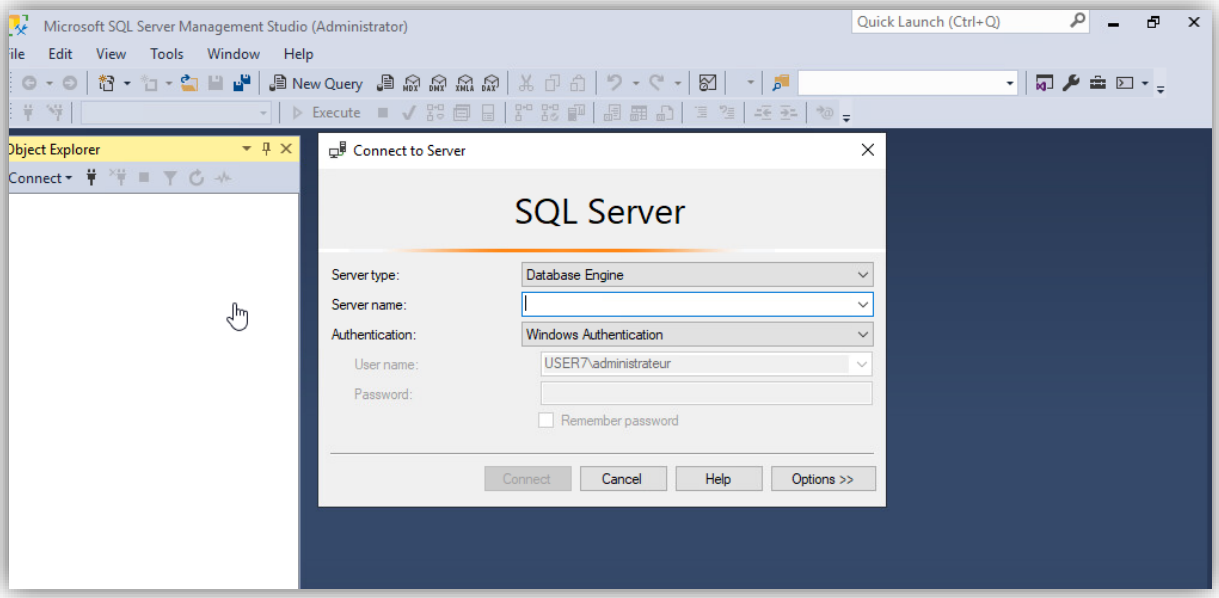


Installation du SQL Server Management Service

Afin de pouvoir me connecter à partir des serveurs, mais aussi à partir du contrôleur de domaine pour effectuer les tests, j'installe l'application client SQL Server Management Service.

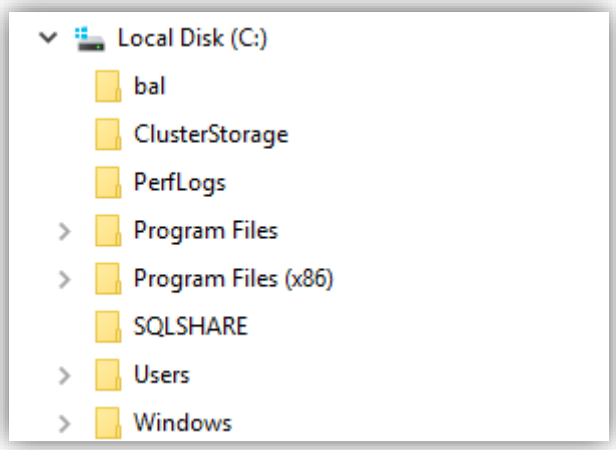




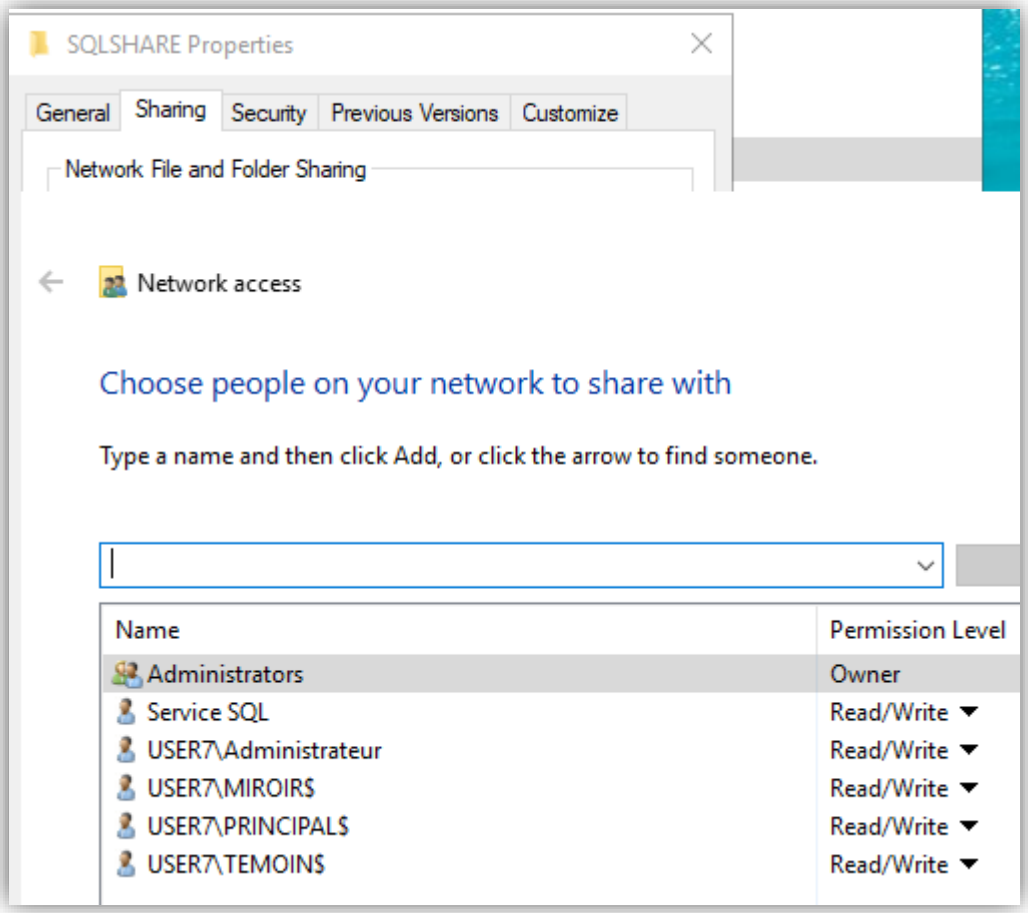


Création d'un partage pour le groupe Always On

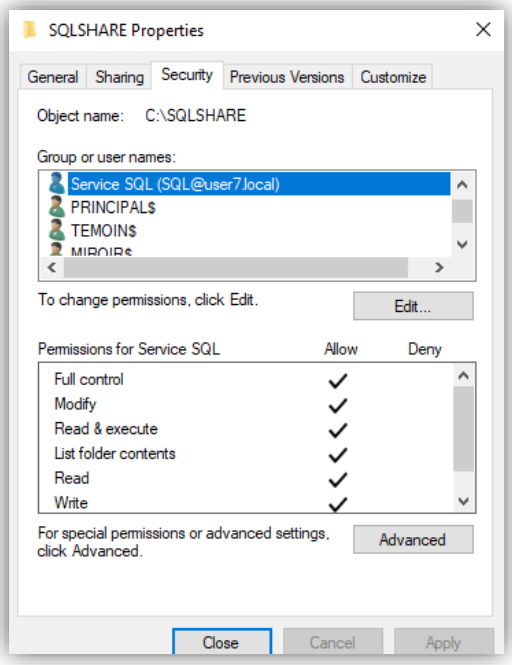
La sauvegarde des bases de données et des fichiers de transactions s'effectuera sur un partage entre les nœuds. Je crée donc un fichier nommé SQLSHARE sur la racine du serveur PRINCIPAL

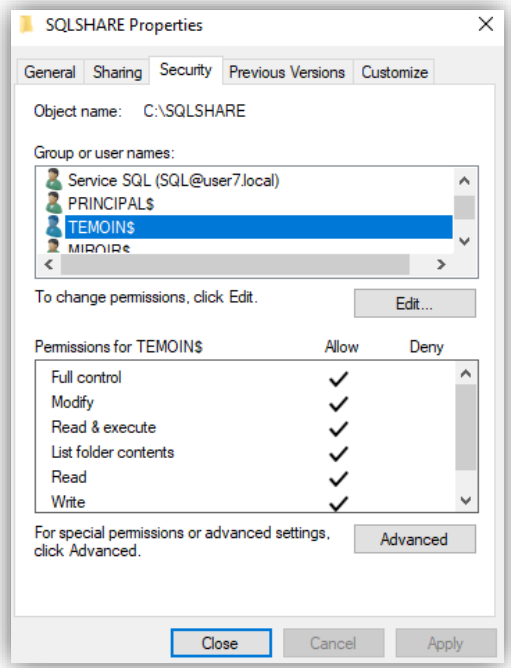


Que je partage avec les ordinateurs qui feront partie du groupe Always On…



Le partage modifie à la fois les permissions de partage et NTFS.

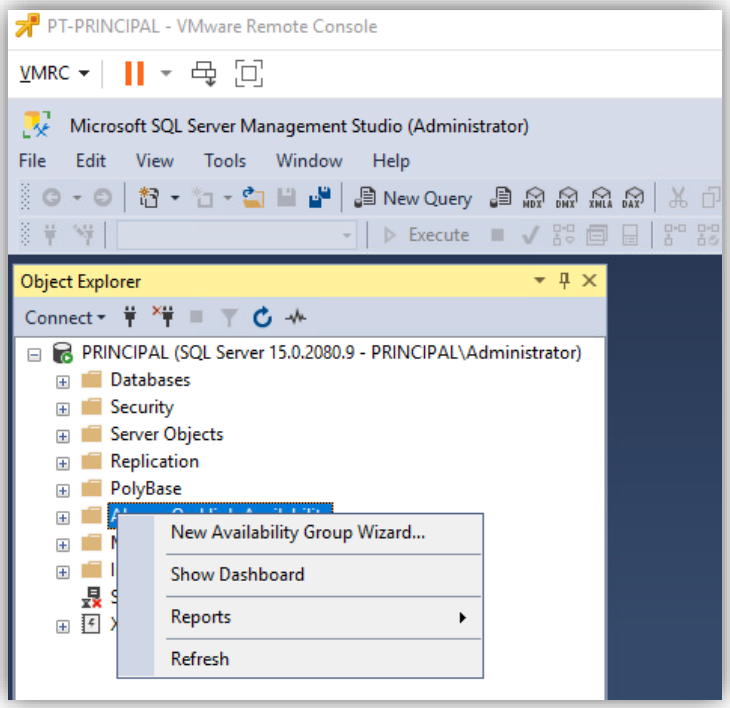




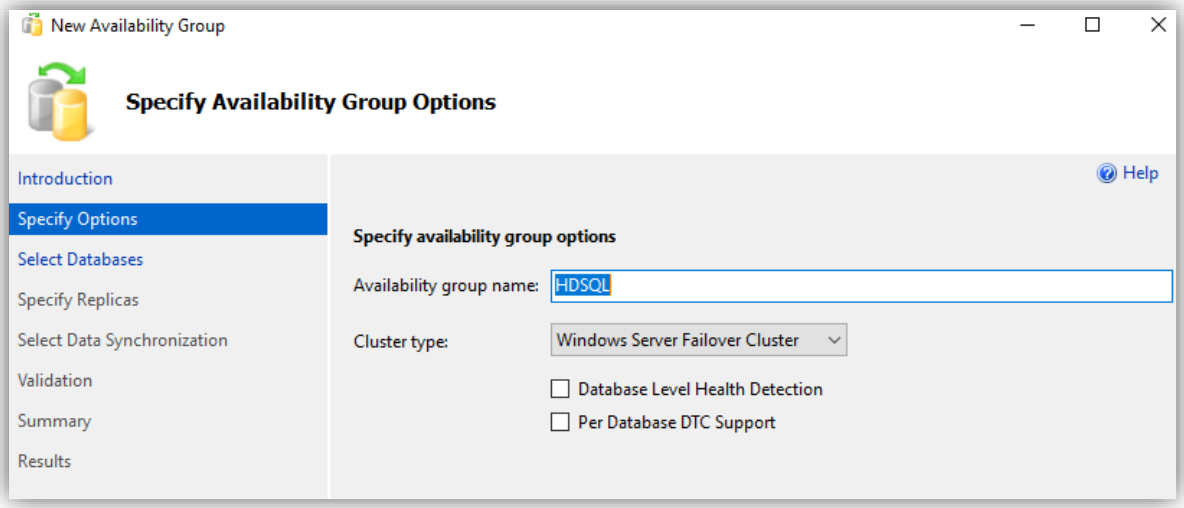
L'infrastructure est prête pour la création du groupe Always On.

Création et configuration du groupe Always On

La création de groupes Always On s'effectue grâce à l'assistant intégré à parti d'un clic droit sur « goupoes de heute disponibilité »

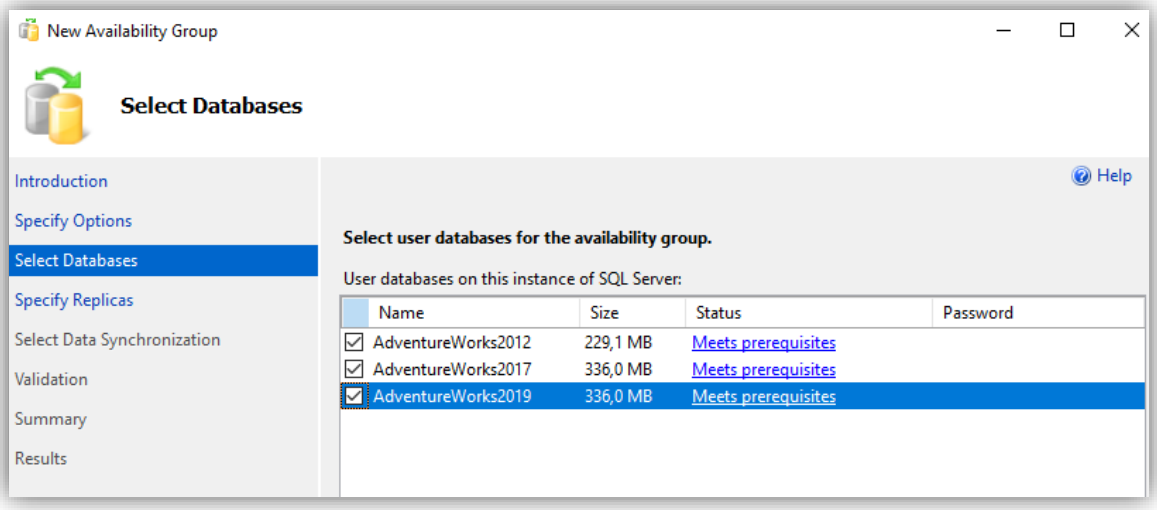


On choisit le nom du groupe. Il est possible d'ajouter une détection automatique d'intégrité de la base de données (Database Level Health detection). Le service optionnel DTC permet de gérer les transactions « distribuées » qui portent sur plusieurs bases de données. Les deux fonctionnalités ne sont pas nécessaires dans ce cadre.



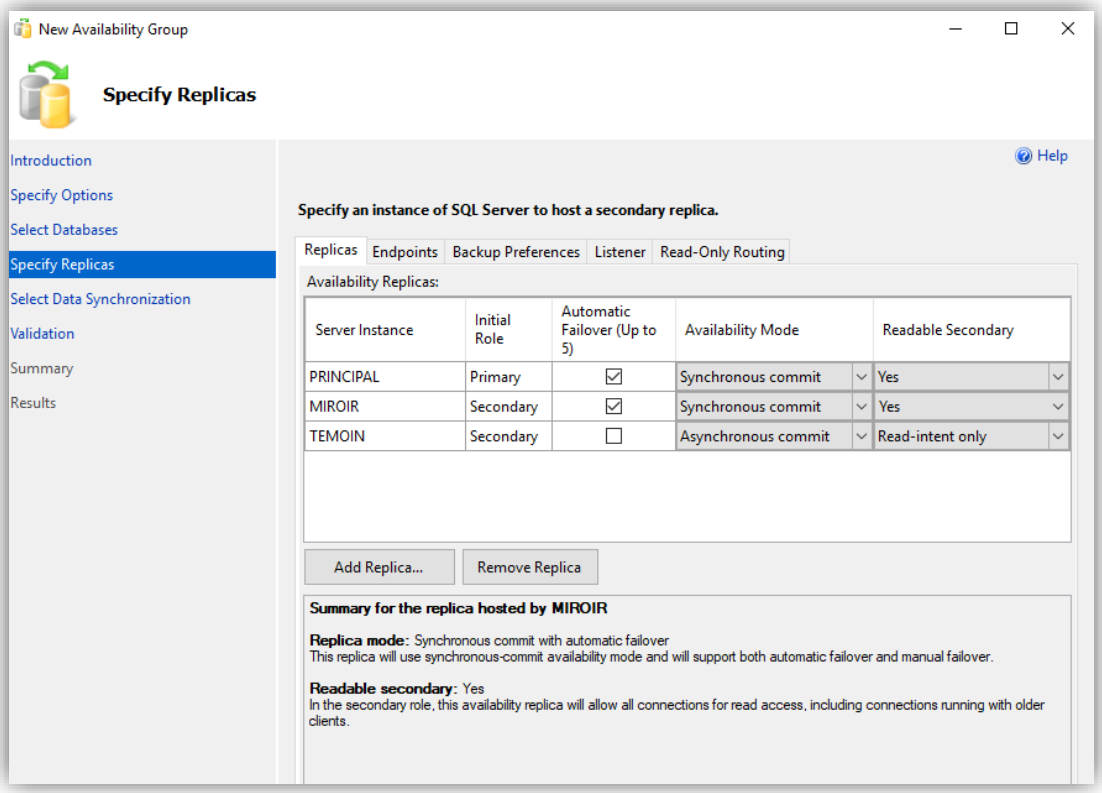
On choisit par la suite les bases de données que l'on souhaite intégrer dans le groupe. Pour faire partie d'un groupe de haute disponibilité, les bases de données doivent être en mode FULL RECOVERY, et une sauvegarde complète doit exister.

Les bases de données suivantes remplissent ces conditions.

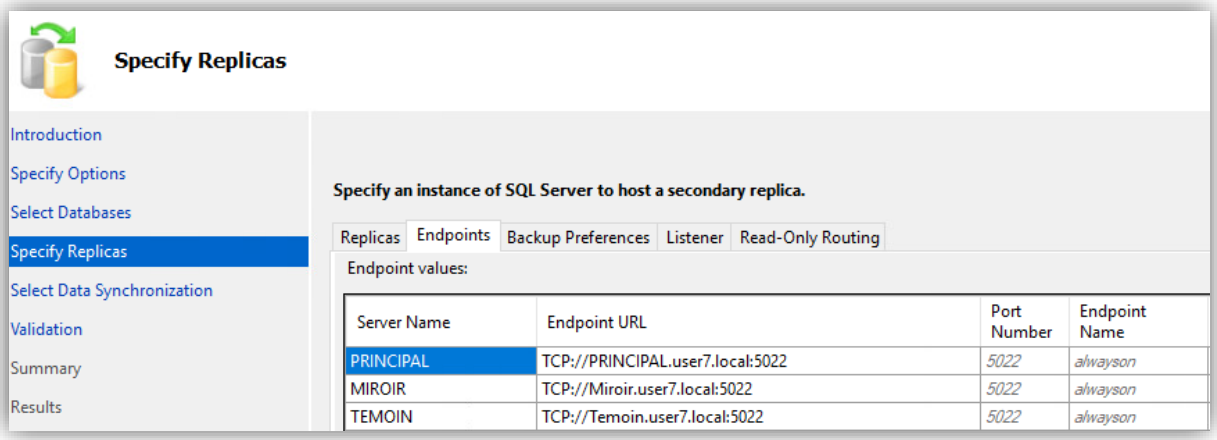


Le choix suivant porte sur les serveurs secondaires qui feront partie du groupe. On choisit également le mode de disponibilité ; la synchronisation des transactions peut s'effectuer de façon synchrone ou asynchrone comme lors de la mise en miroir. Lors du mode synchronous-commit, le serveur principal attend que le serveur secondaire confirme la bonne réception des journaux de transactions avant d'enregistrer définitivement les transactions lui-même. Ceci empêche la perte de données mais augmente la latence. Lors du mode asynchrone, le principal confirme ses transactions aux clients connectés sans attendre que le secondaire ait reçu les journaux de transaction. Ceci améliore la performance au prix de la garantie de l'intégrité des données.

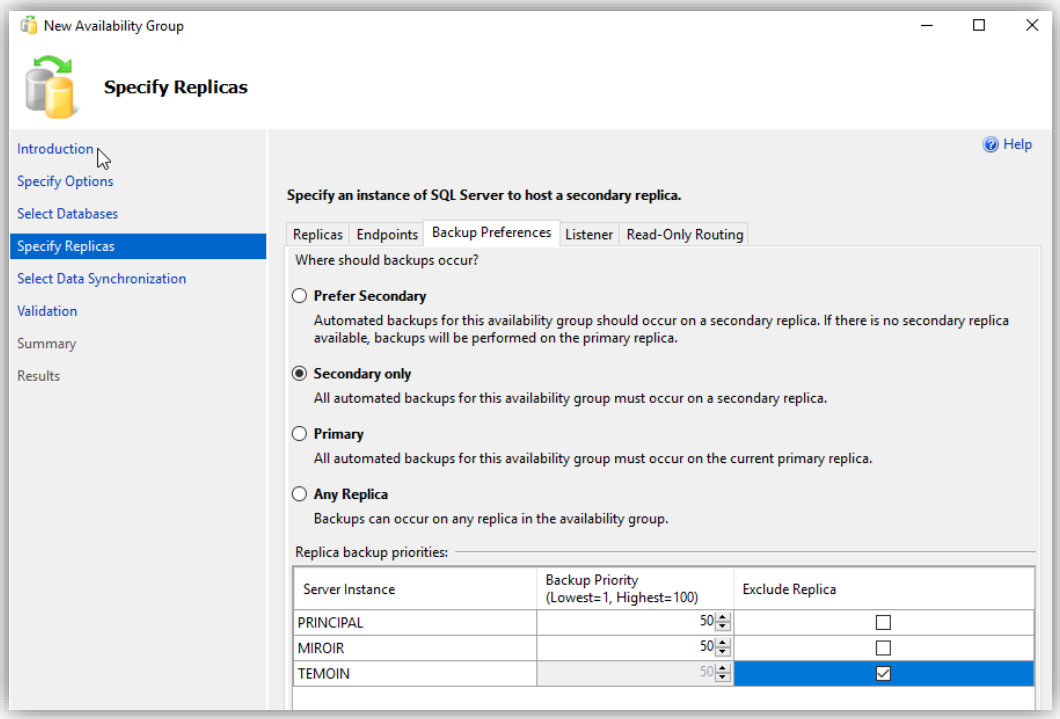
Seul le mode synchronous commit permet le basculement automatique.



L'assistant crée alors de points de terminaison par lesquels seront copiées les transactions. Le port par défaut est 5022. Par défaut, les points de terminaison s'appellent « hadr_endoint » (high availability disaster recovery), mais je les ai renommés en alwayson.

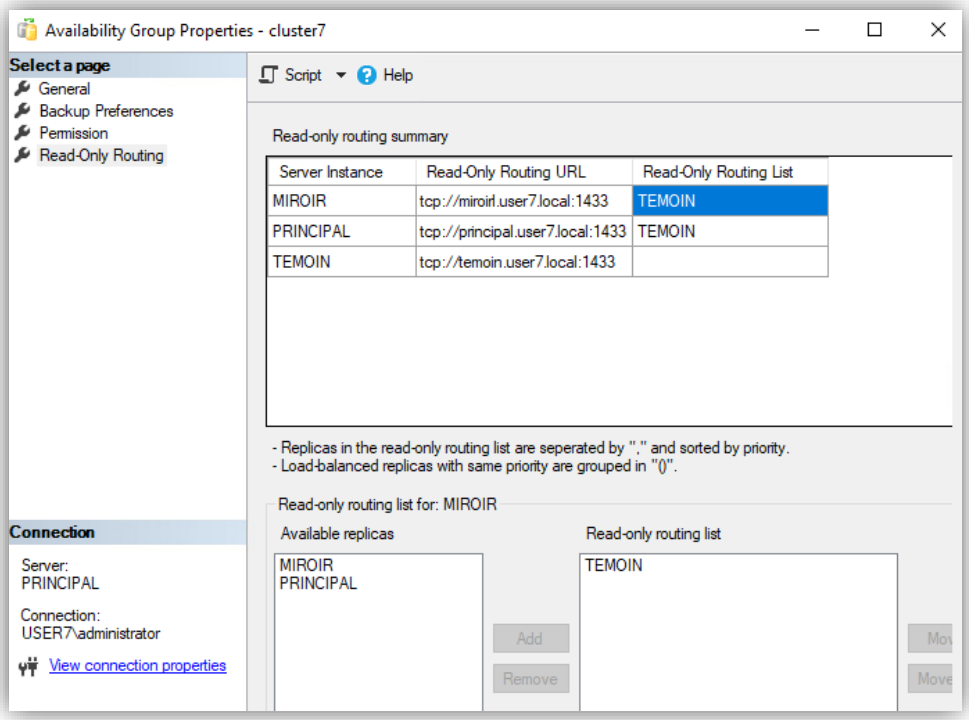


Dans l'onglet « Backup preferences », on définit quels serveurs serviront à la sauvegarde. Le mode par défaut est « Prefer Secondary », mais le mode choisi ici permet de diminuer la charge du principal en sauvegardant uniquement sur les secondaires.

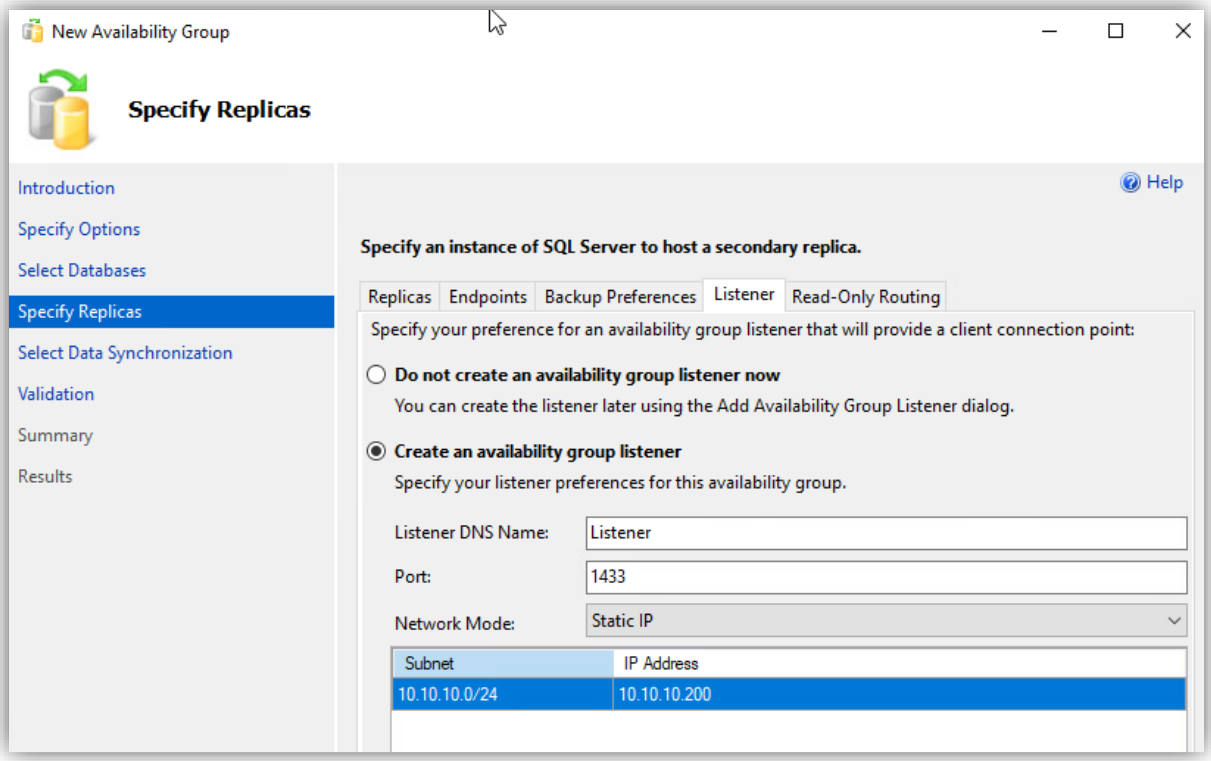


Le Read-Only Routing permet de rediriger les requêtes en lecture seule sur un ou plusieurs serveurs secondaires. Ceci peut également contribue rà la répartition de charge.

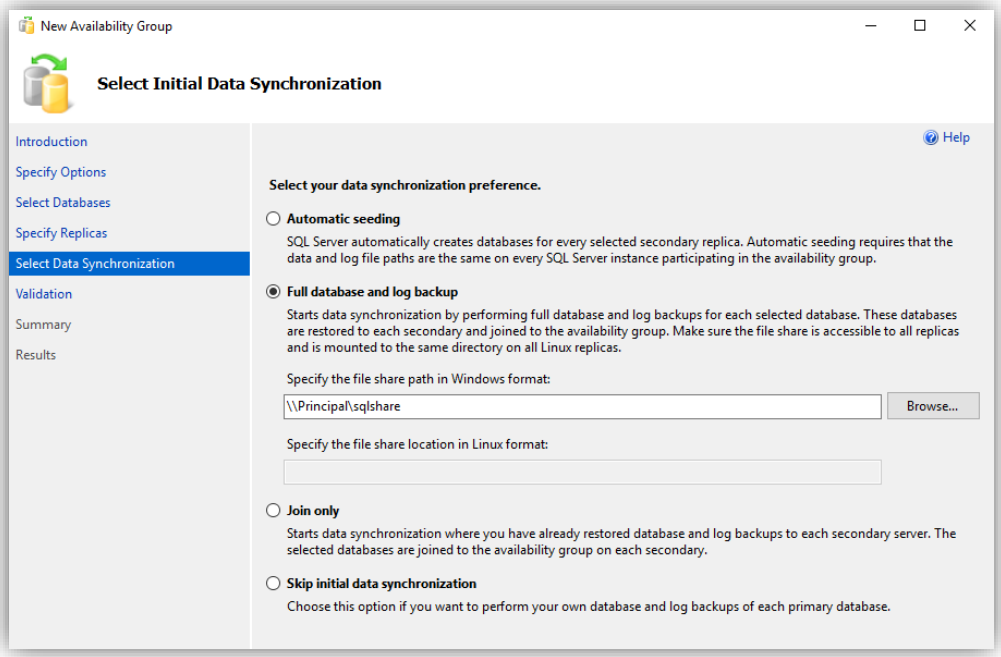
Je configure TEMOIN comme secondaire pour les requêtes en lecture seule



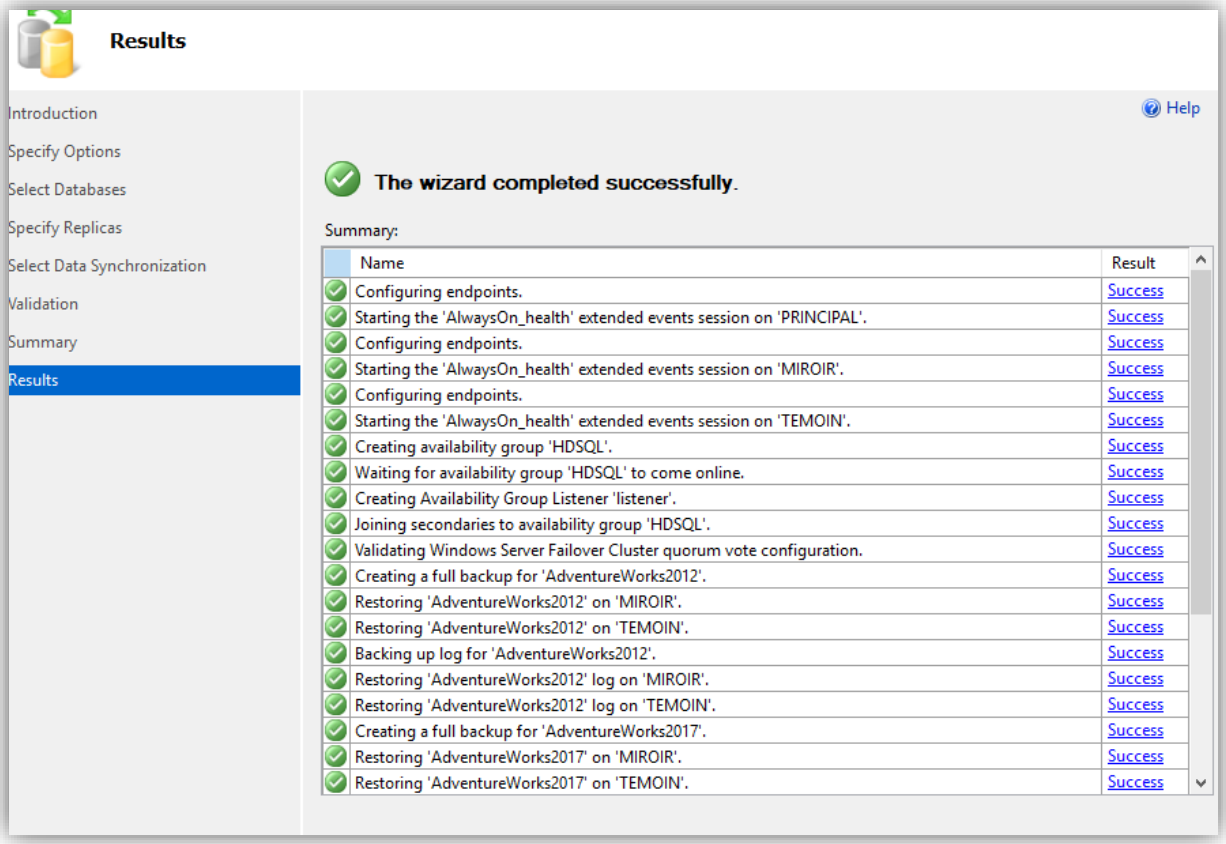
L'assistant crée le « Listener », l'équivalent du nom DNS (Virtual Network Name) du cluster Windows Server qui « représente » le groupe d'ordianateurs sur les réseau. Il réceptionne et redirige les conenxions client de façon transparente. Le listener dispose d'une adresse IP et d'un nom DNS. Le port par défaut est 1433.



Parmi les options de synchronisation initiale de données, je choisis le partage préalablement crée sur le principal.



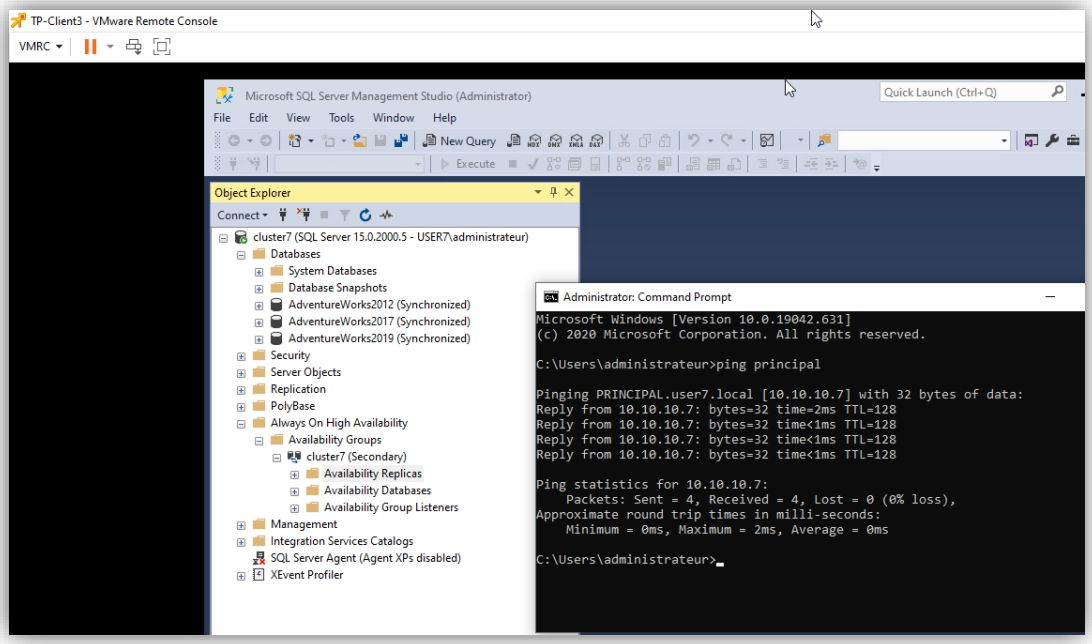
Après validation des paramètres, le groupe de haute disponibilité est à présent opérationnel.

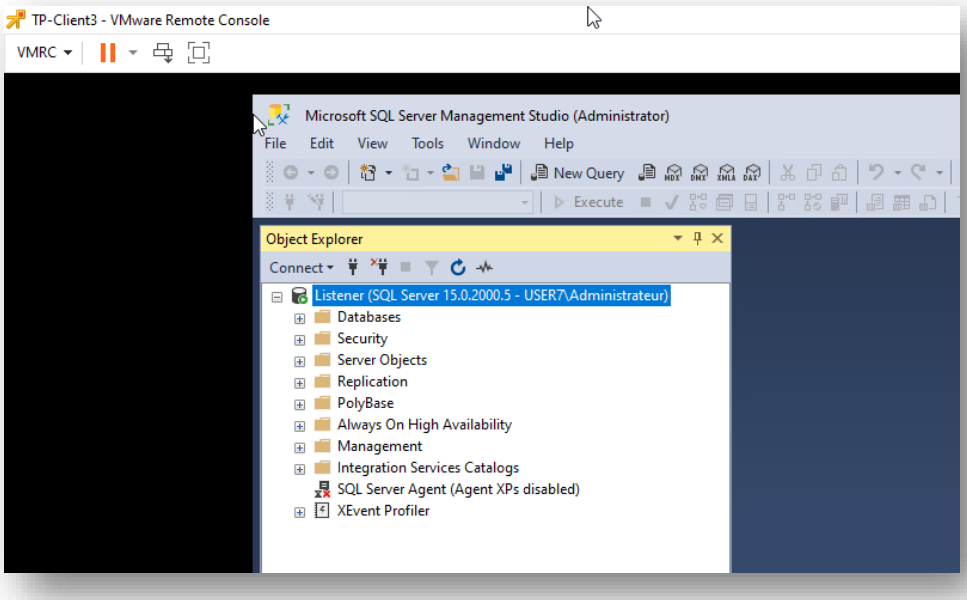


Test de basculement

Je commence par tester l'accès au cluster à partir d'un client Windows 10 qui dispose du SQL Server Management Studio. Tous les serveurs sont en ligne et opérationnels.

Je teste la connexion au cluster par son nom DNS et par le nom DNS du listener

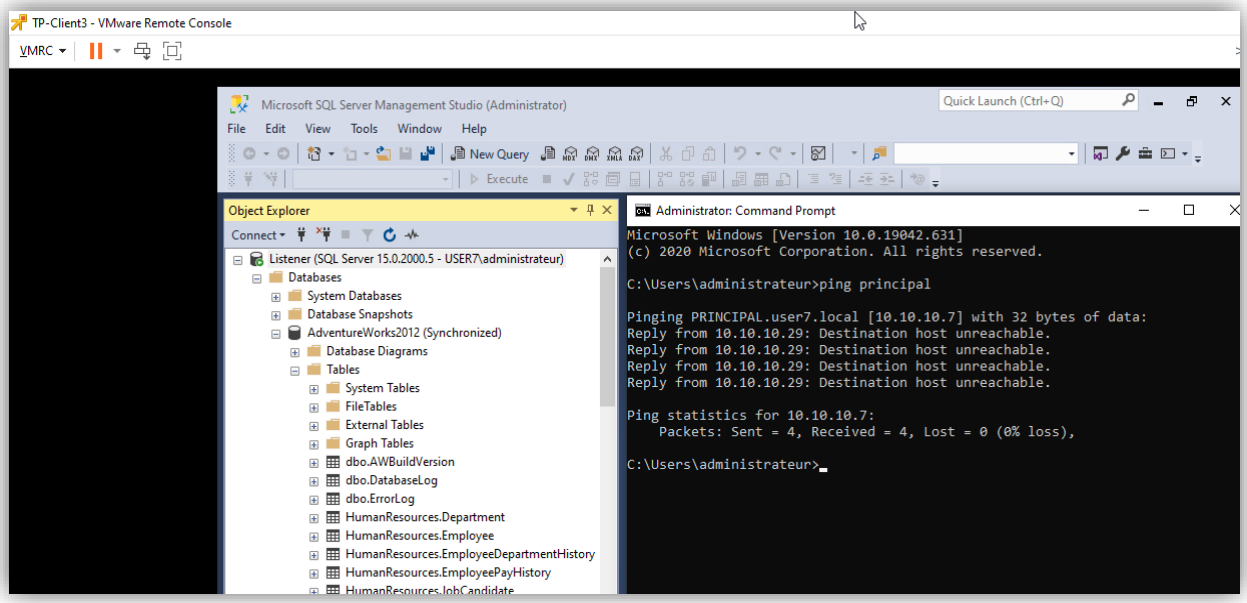




L'accès fonctionne aussi bien par le nom virtuel du cluster que par le nom du listener.

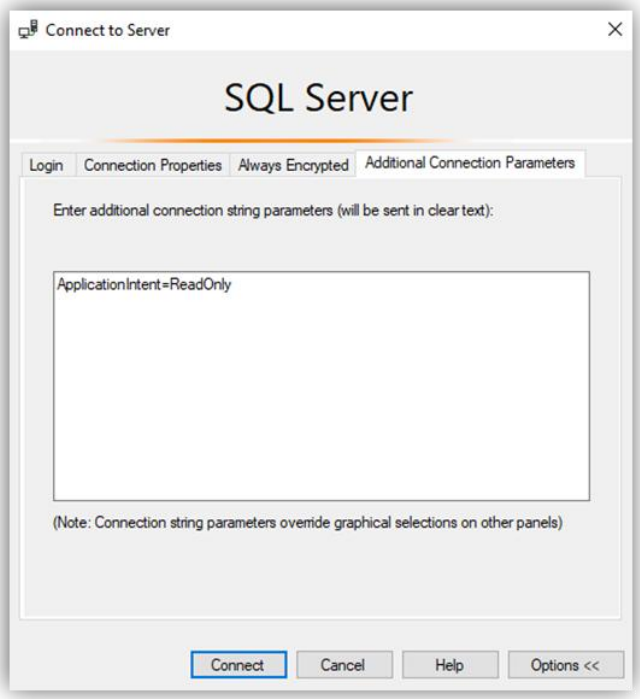
Simulation de panne

Je coupe la connexion réseau du server PRINCIPAL et retente la connexion. Elle réussit.

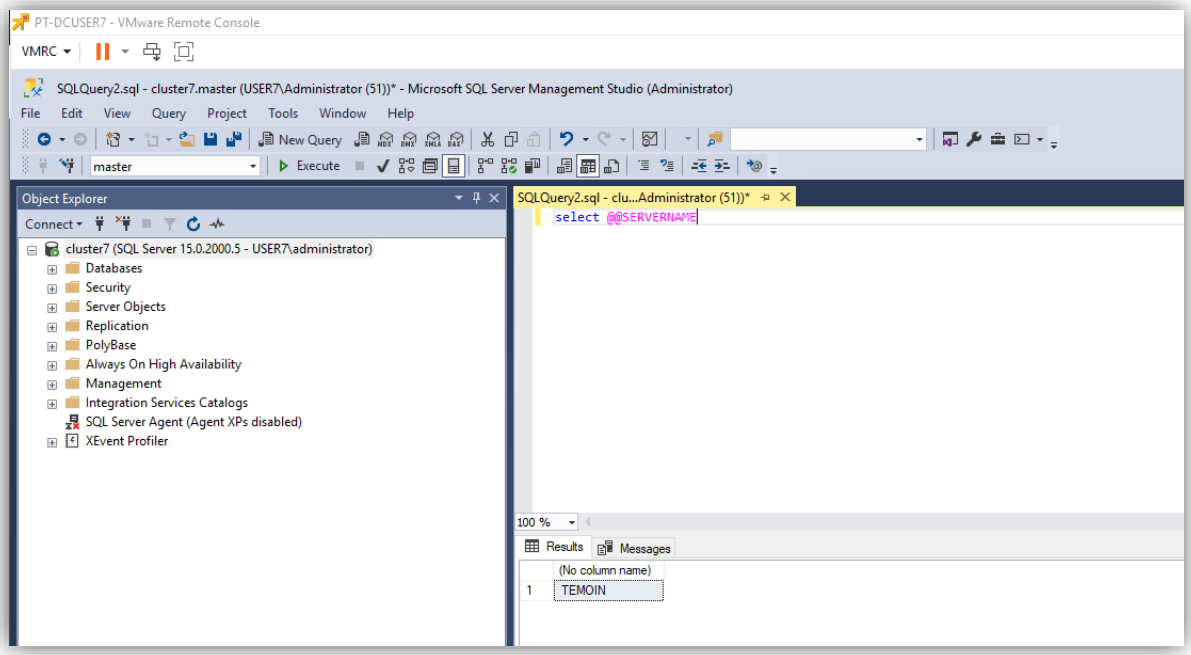


Le Cluster reste donc opérationnel en cas d'interruption du service sur l'instance principale.

Pour tester le read-only routing, je me connecte en utilisant les mêmes paramètres qu'auparavant, mais avec l'option de lecture seule.



La connexion réussit. J'effectue une requête SQL pour connaitre le nom du serveur sur lequel je suis connectée. La requête SQL confirme que la connexion a été transférée au server configuré en lecture seul, Témoin.

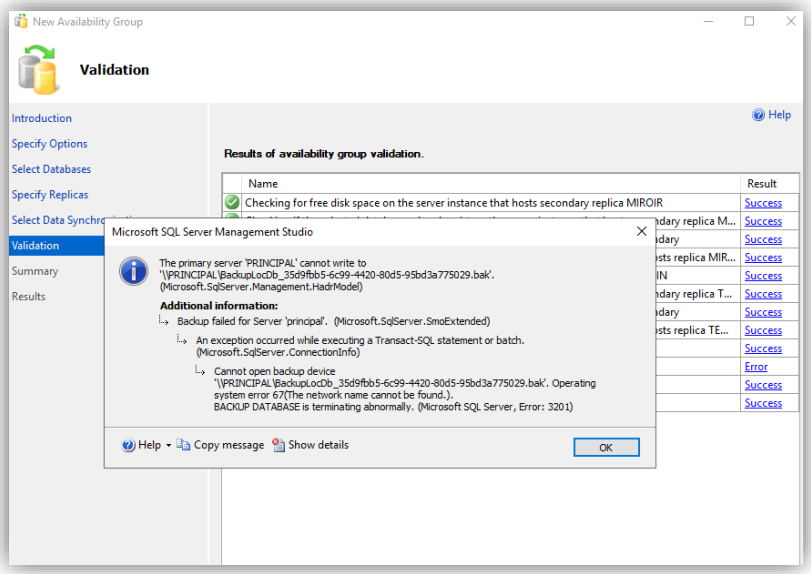


Les tests de basculement se sont terminés avec succès.

Problèmes rencontrés lors de la mise en pratique

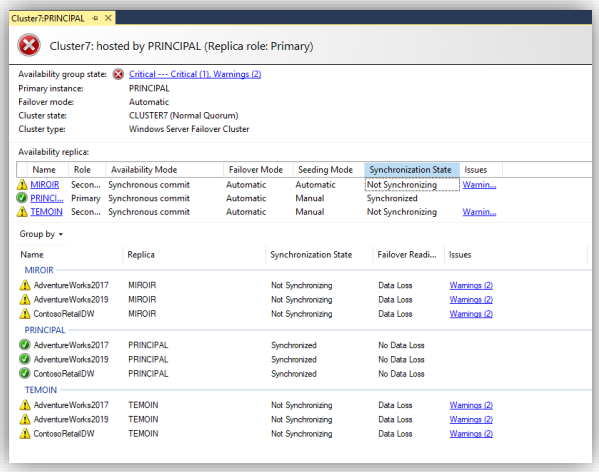
SQL SERVER étant un logiciel complexe, il est facile de commettre des erreurs lors de l'installation et configuration des services. Voici quelques exemples de messages d'erreurs que j'ai pu recevoir et leurs causes respectives :

Impossibilité de se connecter au partage crée pour le groupe :



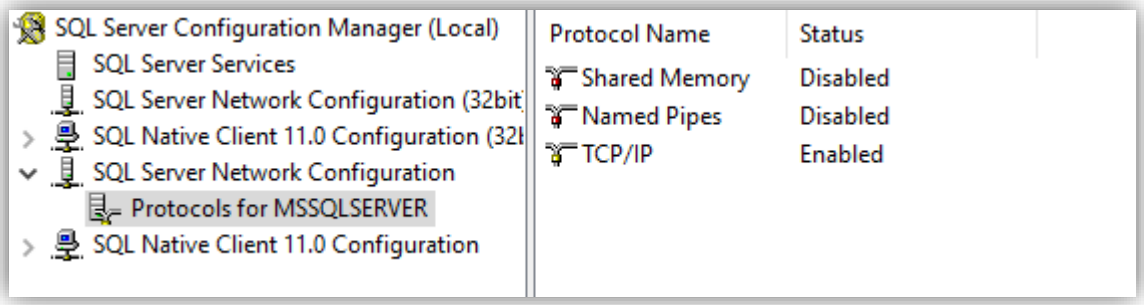
Le fichier partagé sur le bureau ne semblait pas être visible. Curieusement, après avoir déplacé le fichier du bureau à la racine du serveur Principal, l'accès était possible.

Synchronisation impossible :



Après ajout des bases de données au groupe, celles-ci n'étaient pas synchronisées.

Causes : J'avais ouvert les pares-feux pour les ports 5022 et 1433, mais il fallait également configurer les protocoles de communication au niveau du service SQL Server.



Consommation de ressources matérielles :

Pour réaliser ce projet tuteuré, je disposais d'un serveur de virtualisation avec 16 g° de RAM. Bien que le site de Microsoft préconise 4 g° de mémoire, le fonctionnement des machines virtuelles était souvent ralenti, et ce bien que les bases de données n'étaient pas grandes et aucune transaction complexe n'ait été effectuée. Il est nécessaire de prévoir un surplus de ressources matérielles pour garantir un bon fonctionnement.

Conclusion

Le concept Always On présente de nombreux avantages par rapport aux autres solutions de haute disponibilité telles que le Windows Server Failover Cluster, la mise en miroir et la réplication. Le concept de groupes de haute disponibilité en particulier permet de réunir les avantages du Failover clustering et de la mise en miroir :

Il est possible de sécuriser des instances SQL Server entières sans pour autant utiliser un volume partagé.

Si on souhaite appliquer la haute disponibilité au niveau des bases de données, il est possible de regrouper plusieurs bases de données et de les rendre disponibles en hot-standby sur plusieurs serveurs secondaires.

La redirection de requêtes en lecture seule permet une répartition de la charge réseau.

Les Service SQL peut être géré par un groupe d'utilisateurs AD, ce qui facilite nettement la gestion de l'authentification.

Le logiciel Microsoft SQL Server étant très complexe, il est primordial d'appliquer un grand soin à la planification et au choix de la solution de haute disponibilité la plus adaptée. Des erreurs de configuration paraissant minimes peuvent entrainer des dysfonctionnements à large échelle qu'il est difficile de réparer une fois les installations terminées.

Il est également important de retenir que MS SQL Server est « gourmand » en ressources et de prévoir du matériel suffisamment puissant pour supporter un bon fonctionnement.

Sources

Archived docs. “Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery.” *Microsoft Docs*, docs.microsoft.com/en-us/previous-versions/sql/sql-server-2012/hh781257(v=msdn.10).

Author: Syed Yousuf, and Author: “Enable Remote Connections to SQL Server Using IP Address.” *TimeXtender Support*, support.timextender.com/hc/en-us/articles/360042584612-Enable-Remote-Connections-to-SQL-Server-using-IP-address.

“Build a SQL Cluster Lab Part 3.” *TECHCOMMUNITY.MICROSOFT.COM*, 18 Nov. 2019, techcommunity.microsoft.com/t5/core-infrastructure-and-security/build-a-sql-cluster-lab-part-3/ba-p/981679.

Caesar, Daniel, et al. *SQL-Server 2019 für Administratoren Das Umfassende Handbuch*. Rheinwerk Computing, 2020.

Carter, Peter A. “Administering AlwaysOn.” *SQL Server AlwaysOn Revealed*, 2015, pp. 129–147., doi:10.1007/978-1-4842-1763-4_6.

Cawrites. “Configure Windows Firewall - SQL Server.” *Configure Windows Firewall - SQL Server | Microsoft Docs*, docs.microsoft.com/en-us/sql/sql-server/install/configure-the-windows-firewall-to-allow-sql-server-access?view=sql-server-ver15.

Cawrites. “Start Data Movement on a Secondary Database - SQL Server Always On.” *Start Data Movement on a Secondary Database - SQL Server Always On | Microsoft Docs*, docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/start-data-movement-on-an-always-on-secondary-database-sql-server?view=sql-server-ver15.

Carter, Peter A. *SQL Server AlwaysOn Revealed: Supporting 24 x 7 Operations with Continuous Uptime*. Apress, 2015.

Kloep, Peter, et al. *Windows Server 2019 Das Umfassende Handbuch*. Rheinwerk Computing, 2019.

Konopasek, Klemens. *SQL-Server 2014 Der Schnelle Einstieg*. Hanser, 2014.

Les Clés De L'IT : Spécial Disaster Recovery, www.bitpipe.fr/fulfillment/1518178989_625.

“MS SQL Server - HA Technologies.” *Tutorialspoint*, www.tutorialspoint.com/ms_sql_server/ms_sql_server_ha_technologies.htm.

Parui, Uttam. *Pro SQL Server Always on Availability Groups*. Apress, 2016.

SerdarSoysal. “Configure SQL Server AlwaysOn Availability Groups for SharePoint Server - SharePoint Server.” *SharePoint Server | Microsoft Docs*, docs.microsoft.com/en-us/sharepoint/administration/configure-an-alwayson-availability-group.

Sheldon, Robert. “Von AlwaysOn Bis Replikation: Verbesserte Hochverfügbarkeit Mit SQL Server 2014.” *ComputerWeekly.de*, ComputerWeekly.com/De, 24 Apr. 2014, www.computerweekly.com/de/tipp/Von-AlwaysOn-bis-Replikation-Verbesserte-Hochverfuegbarkeit-mit-SQL-Server-2014.

“SQL Server AlwaysOn Availability Groups Einrichten Von Holger Voges.” *PDF Free Download*, docplayer.org/666936-Sql-server-alwayson-availability-groups-einrichten-von-holger-voges.html.

Upton, Brady. “SQL Server AlwaysOn Availability Groups - Part 1 Configuration.” *SQL Server Tips, Techniques and Articles*, MSSQLTips, 8 Nov. 2011, www.mssqltips.com/sqlservertip/2519/sql-server-alwayson-availability-groups--part-1-configuration/.

Ward, Bob. “High Availability and Disaster Recovery for SQL Server.” *Pro SQL Server on Linux*, 2018, pp. 369–436., doi :10.1007/978-1-4842-4128-8_8.