Editions ENI

PowerShell, Hyper-V et SCVMM

Administrez et orchestrez votre infrastructure virtuelle avec PowerShell

Collection
Expert IT

Extrait

Chapitre 4
Les clusters d'hôtes

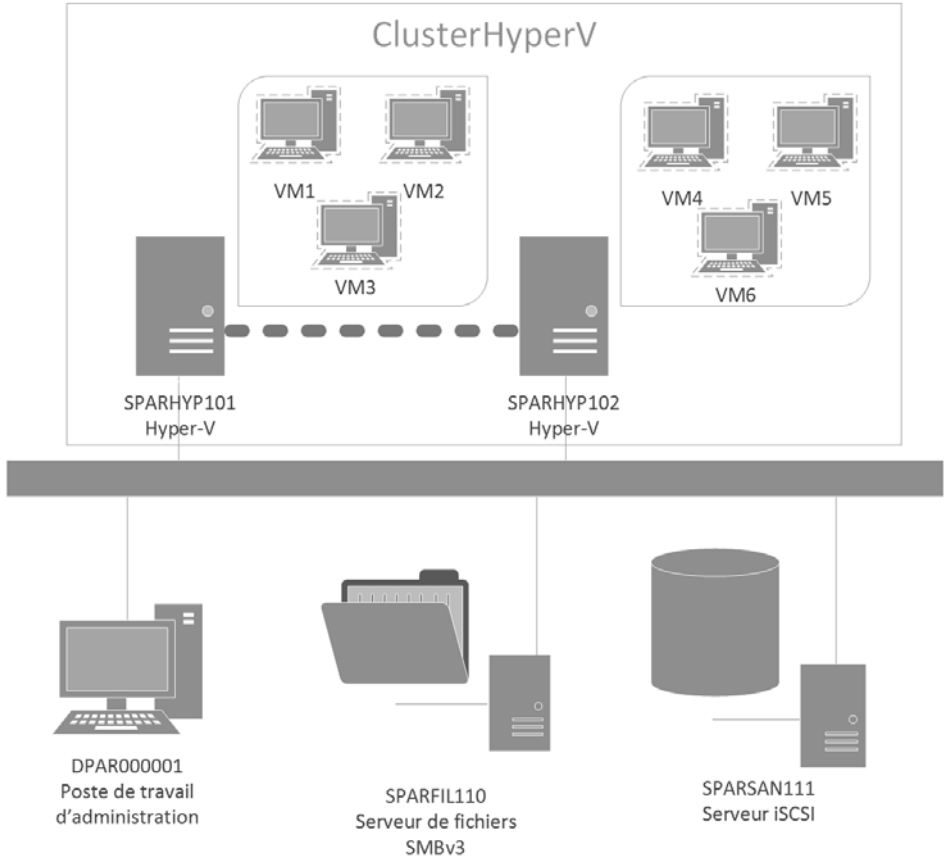
1. Objectif

L'objectif de ce chapitre est de capitaliser les connaissances acquises lors du chapitre précédent sur la gestion de Hyper-V Server avec PowerShell, et de mettre en œuvre un cluster d'hôtes Hyper-V. Pour le stockage, un volume CSV et un partage SMBv3 seront utilisés. Tous deux supportent la haute disponibilité pour les machines virtuelles. L'architecture réseau introduit d'une part l'agrégation de cartes (teaming) et d'autre part la convergence réseau où le commutateur virtuel devient le point d'entrée.

Les améliorations et impacts sur la gestion des machines virtuelles sont ensuite abordés, avec la haute disponibilité, la migration et la réplication de machines virtuelles.

Le chapitre finit en présentant la mise à jour des hôtes du cluster avec la fonctionnalité de mise à jour adaptée aux clusters, ou Cluster-Aware Updating.

L'architecture de la plateforme utilisée dans ce chapitre est représentée par le schéma ci-dessous.



Cette fois les deux hôtes `SPARHYP101` et `SPARHYP102` sont mis en cluster, pour le stockage, ils sont connectés au serveur de fichiers `SPARFIL110` avec le protocole SMBv3 et au serveur `SPARSAN111` qui propose des LUNs iSCSI.

2. Cluster d'hôtes

2.1 Configuration du stockage

Dans ce chapitre, le stockage local à un hôte n'est plus utilisé, seul le stockage distant sera utilisé ; soit avec le protocole SMBv3 et des partages Windows, soit en utilisant des LUNs proposés par un SAN.

L'utilisation de partages et SMBv3 ne requiert aucune configuration particulière, il faut que le serveur de fichier supporte cette version de protocole. Cette partie a été présentée dans le chapitre précédent cf. PowerShell, Hyper-V et Windows Server 2016 - Configuration du stockage et n'est pas reprise ici.

L'utilisation de LUN a aussi été vue dans le chapitre précédent, la différence ici, est que tous les hôtes sont configurés pour se connecter aux mêmes LUNs. Il est donc nécessaire de créer des volumes CSV sur les LUNs. Cette opération est présentée après la mise en cluster des hôtes.

Dans le cas de la plateforme proposée, deux LUN seront utilisés, la première de 1 Go servira à stocker le quorum du cluster, l'autre servira à héberger les machines virtuelles.

2.2 Configuration du réseau

Le chapitre précédent a mis en œuvre des commutateurs virtuels de manière simple en utilisant une carte réseau par commutateur. Dans les faits, les serveurs possèdent plusieurs cartes réseau utilisées pour différentes fonctions : le réseau des machines virtuelles, de gestion, pour le stockage SMB ou iSCSI, pour le cluster, la migration de machines virtuelles, la réplication.

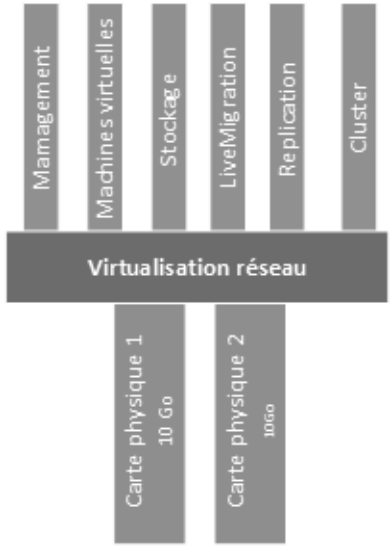
Pour assurer la haute disponibilité du réseau, les cartes sont couplées dans un agrégat ou teaming de cartes, cela double le nombre de cartes réseau nécessaires. De nos jours, l'équipement des serveurs a tendance à réduire le nombre de cartes réseau, au profit de cartes 10 Gbits/sec au lieu de 1 Gbits/sec. Une configuration courante de serveur est de posséder deux cartes 10 Gbits/sec et deux cartes 1 Gbits/sec. La convergence réseau apporte une solution, elle consiste à utiliser un commutateur virtuel et redéfinir des cartes réseau virtuelles pour accéder aux différents réseaux. Hyper-V 2012 proposait une architecture en créant un teaming de cartes, puis un switch virtuel sur lequel sont récrés les différents réseaux nécessaires. Hyper-V 2016 va plus loin en prenant en charge l'équilibrage de charge au sein du commutateur virtuel, ce qui dispense de créer un agrégat de cartes.

Le tableau ci-dessous décrit les différents types de réseau nécessaires sur une infrastructure Hyper-V.

Réseau	Description
Management	Réseau de communication entre les hôtes Hyper-V, avec l'infrastructure, DNS, Active Directory … Réseau d'administration des serveurs.
Cluster	Connectivité intracluster ; heartbeat.
Machines virtuelles	Réseaux utilisés par les machines virtuelles pour communiquer entre elles et avec le monde extérieur : entreprise, internet.
Stockage	Réseau utilisé par le stockage dans le cas de SMB ou iSCSI.
Live Migration	Réseau utilisé pour la migration de machines virtuelles.
Réplication	Réseau utilisé pour la réplication des machines virtuelles

Les anciennes topologies de réseau utilisent un teaming de deux cartes réseau pour chaque réseau nécessaire, ce qui nécessite 12 cartes physiques si tous les réseaux sont séparés. En dehors du problème de coût et de la gestion de la connectivité, cette solution n'est pas optimum, car certains réseaux consomment peu de bande passante par rapport à d'autres, il n'y a pas de répartition de charge entre ces différents réseaux puisqu'ils sont physiquement séparés. Utiliser un switch virtuel et redéfinir les cartes réseau sur ce commutateur permet d'optimiser la bande passante car le teaming va répartir la charge sur toutes les cartes. De plus, le commutateur virtuel supporte la gestion de la qualité de service, et l'ajout d'un poids à chaque réseau apporte l'assurance que la bande passante ne sera pas entièrement utilisée par l'un d'entre eux. L'évolution matérielle va aussi dans le sens de la convergence réseau, il est préférable d'utiliser deux cartes réseau à 10 Gbits/sec, plutôt que quatre ou six cartes à 1 Gbits/sec.

La logique actuelle est représentée par le schéma suivant : les cartes physiques sont agrégées et la couche de virtualisation réseau redéfinit une nouvelle topologie réseau. Les cartes de 1 Gb ne sont pas utilisées ou dédiées à un réseau, mais ne font pas partie de l'agrégat.

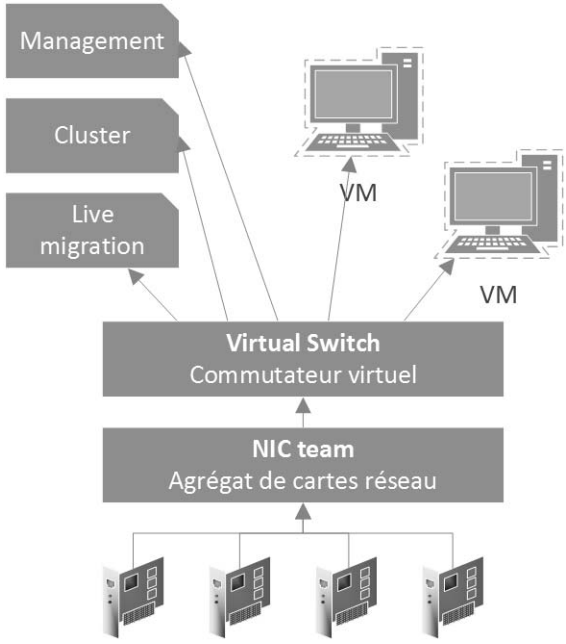


Cela revient à utiliser une autoroute pour construire plusieurs routes départementales à travers un échangeur. Le flux global est optimisé par l'autoroute, les péages (gestion de qualité) vont réguler le débit des routes départementales.

Il y a aujourd'hui deux manières de créer l'échangeur, utiliser un agrégat de cartes et un commutateur virtuel, ou uniquement un commutateur virtuel, ces deux solutions sont présentées dans les sections suivantes.

2.2.1 Commutateur virtuel sur un agrégat de cartes

La première proposition, dont le schéma est présenté ci-dessous, assure la haute disponibilité du réseau en créant un agrégat de cartes réseau. Le switch virtuel est créé par-dessus à partir du teaming. Des cartes virtuelles sont ensuite créées pour les différents réseaux.



Sur ce schéma, il y a une carte réseau **Management**, **Cluster** et **LiveMigration**, et les réseaux de machines virtuelles.

Editions ENI

PowerShell Core et Windows PowerShell

Les fondamentaux du langage

(2e édition)

Collection
Expert IT

Extrait

Chapitre 12
Sécurité

1. La sécurité : pour qui ? Pourquoi ?

L'arrivée des réseaux locaux et d'Internet a changé beaucoup de choses dans la manière de protéger son PC. Il ne suffit plus d'attacher son disque dur au radiateur et de fermer la porte du bureau le soir pour ne pas se faire voler ou pirater des données. Maintenant, protéger son poste de travail est devenu essentiel pour ne pas faire les frais d'intrusions ou de malveillances.

Mais alors contre qui se prémunir ? Eh bien, contre tout ce qui bouge… et même ce qui ne bouge pas. En effet, que ce soient des programmes malveillants, des utilisateurs mal intentionnés, voire des utilisateurs inexpérimentés, tous peuvent être considérés comme une menace. C'est pour cela que vous devez verrouiller votre système en établissant des règles de sécurité, en les appliquant et en vous assurant que les autres en font tout autant.

2. Les risques liés au scripting

Vous allez vite deviner que ce qui fait la force du scripting en fait aussi sa faiblesse. La facilité avec laquelle vous pouvez tout faire, soit en cliquant sur un script, soit en l'exécutant depuis la fenêtre de commande, peut vous mettre dans l'embarras si vous ne faites pas attention.

Imaginez un script de logon qui dès l'ouverture de la session la verrouille aussitôt ! Alors oui, c'est sympa entre copains, mais en entreprise, nous doutons que cela soit de bon ton. Plus grave encore, un script provenant d'une personne mal intentionnée ou vraiment peu expérimentée en PowerShell (dans ce cas, nous vous conseillons de lui acheter un exemplaire de ce livre…) peut parfaitement vous bloquer des comptes utilisateurs dans Active Directory, vous formater un disque, vous faire rebooter sans cesse. Vous l'avez compris, un script peut tout faire. Car même si aujourd'hui des alertes sont remontées jusqu'à l'utilisateur pour le prévenir de l'exécution d'un script, elles ne sont pas capables de déterminer à l'avance si un script est nuisible au bon fonctionnement du système.

Les risques liés au scripting se résument à une histoire de compromis : soit vous empêchez toute exécution de scripts, c'est-à-dire encourir le risque de vous « pourrir la vie » à faire et à refaire des tâches basiques et souvent ingrates, soit vous choisissez d'ouvrir votre système à PowerShell, en prenant soin de prendre les précautions qui s'imposent.

Mais ne vous laissez pas démoraliser car même si l'exécution de scripts vous expose à certains problèmes de sécurité, PowerShell se dote de certains concepts qui font de lui l'un des langages de script les plus sûrs. Il ne faut pas non plus oublier qu'en cas de problème de sécurité, c'est l'image tout entière de Microsoft qui en pâtit…

3. Optimiser la sécurité PowerShell

3.1 La sécurité PowerShell par défaut

Vous l'avez compris, la sécurité est une chose très importante, surtout dans le domaine du scripting. C'est pour cela que les créateurs de PowerShell ont inclus deux règles de sécurité par défaut.

Des fichiers ps1 associés au bloc-notes

L'extension « **`.ps1`** » des scripts PowerShell, est par défaut associée à l'éditeur de texte bloc-notes (ou Notepad). Ce procédé permet d'éviter de lancer des scripts potentiellement dangereux sur une mauvaise manipulation. Le bloc-notes est certes un éditeur un peu classique, mais il a le double avantage d'être inoffensif et de ne pas bloquer l'exécution d'un script lorsque celui-ci est ouvert avec l'éditeur. Vous remarquerez cependant que l'édition (clic droit + **Modifier**) des fichiers **`.ps1`** est associée à l'éditeur ISE.

Remarque

Ce type de sécurité n'était pas mis en place avec les scripts VBS dont l'ouverture était directement associée au Windows Script Host.

Une stratégie d'exécution restreinte

La seconde barrière de sécurité est l'application de la stratégie d'exécution **`Restricted`** par défaut pour les postes clients et **`RemoteSigned`** celle par défaut pour les serveurs (depuis Windows Server 2012 R2).

La stratégie **`Restricted`** est la plus restrictive. C'est-à-dire qu'elle bloque systématiquement l'exécution de tout script. Seules les commandes tapées dans le shell seront exécutées. Pour remédier à l'inexécution des scripts, PowerShell requiert que l'utilisateur change le mode d'exécution avec la commande **`Set-ExecutionPolicy <mode d'exécution>`**. Mais pour ce faire il faut être administrateur de la machine.

Remarque

Peut-être comprenez-vous mieux pourquoi l'utilisation de PowerShell sur vos machines ne constitue pas un accroissement des risques, dans la mesure où certaines règles sont bien respectées.

3.2 Les stratégies d'exécution

PowerShell intègre un concept de sécurité que l'on appelle les stratégies d'exécution (execution policies) pour qu'un script non autorisé ne puisse pas s'exécuter à l'insu de l'utilisateur. Il existe sept configurations possibles : **`Restricted`**, **`RemoteSigned`**, **`AllSigned`**, **`UnRestricted`**, **`Bypass`**, **`Default`** et **`Undefined`**. À chacune d'elles correspond un niveau d'autorisation d'exécution de scripts particulier. Vous pourrez être amené à en changer en fonction de la stratégie que vous souhaitez appliquer.

3.2.1 Les différentes stratégies d'exécution

`Restricted` : c'est la stratégie la plus restrictive, et c'est aussi la stratégie par défaut sur les postes clients (de Windows 7 à Windows 10). Elle ne permet pas l'exécution de scripts mais autorise uniquement les instructions en ligne de commande tapées dans la console (mode interactif). Cette stratégie peut être considérée comme la plus radicale étant donné qu'elle protège contre l'exécution des fichiers **`.ps1`**.

Lors d'une tentative d'exécution de script avec cette stratégie, un message de ce type est affiché dans la console :

```
.\lanceur.ps1 : File C:\temp\lanceur.ps1 cannot be loaded because run-
ning scripts is disabled on this system. For more information, see
about_Execution_Policies at https://go.microsoft.com/fwlink/
?LinkID=135170.
At line:1 char:1
+ .\lanceur.ps1
+ ~~~~~~~~~~~~~
```

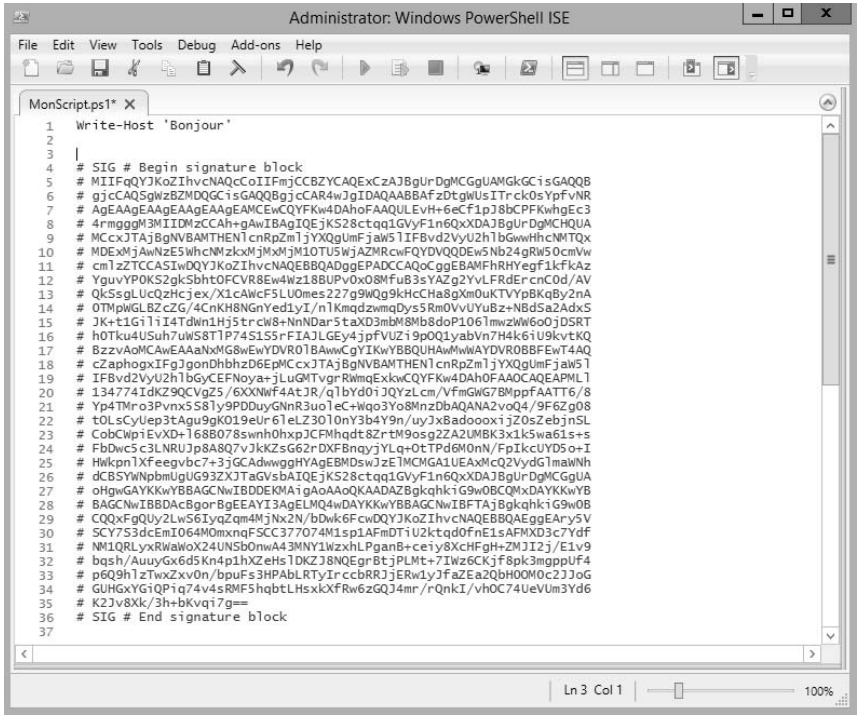
```
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

Si cette stratégie est celle définie par défaut lors de l'installation de PowerShell, il vous faudra la changer pour l'exécution de votre premier script.

Remarque

Pour pouvoir modifier la stratégie d'exécution PowerShell, il vous faudra être administrateur local de la machine.

`AllSigned` : c'est la stratégie permettant l'exécution de scripts la plus « sûre ». Elle autorise uniquement l'exécution des scripts signés. Un script signé est un script comportant une signature numérique comme celle présentée sur la figure ci-dessous.



Exemple de script signé

Avec la stratégie **`AllSigned`**, l'exécution de scripts signés nécessite que vous soyez en possession des certificats correspondants (cf. section Signature des scripts).

RemoteSigned : cette stratégie se rapporte à **AllSigned** à la différence près que seuls les scripts ayant une origine autre que locale nécessitent une signature. Par conséquent, tous vos scripts créés localement peuvent être exécutés sans être signés.

À partir de Windows Server 2012 R2, PowerShell s'exécute désormais avec cette stratégie d'exécution par défaut, ce qui n'était pas le cas dans les versions antérieures de Windows Server.

Si vous essayez d'exécuter un script provenant d'Internet sans que celui-ci ne soit signé, vous obtiendrez le message d'erreur suivant :

```
.\Script.ps1 : File C:\Temp\Script.ps1 cannot be loaded.
The file C:\Temp\Script.ps1 is not digitally signed.
```

Remarque

Vous vous demandez sûrement comment PowerShell fait pour savoir que notre script provient d'Internet ? Réponse : Grâce aux « Alternate Data Streams » qui sont implémentés sous forme de flux cachés depuis des applications de communication telles que Microsoft Outlook, Internet Explorer, Outlook Express et Windows Messenger (voir partie traitant des Alternate Data Streams). En gros, lorsqu'un script est téléchargé à partir d'un client Microsoft, la provenance de celui-ci lui est attachée.

Unrestricted : avec cette stratégie, tout script, peu importe son origine, peut être exécuté sans demande de signature.

Cette stratégie affiche tout de même un avertissement lorsqu'un script téléchargé d'Internet tente d'être exécuté.

```
PS > .\script.ps1

Security warning

Run only scripts that you trust. While scripts from the internet can
be useful, this script can potentially harm your computer. If you trust
this script, use the Unblock-File cmdlet to allow the script to run
without this warning message. Do you want to run C:\Temp\script.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"):
```

Bypass : c'est la stratégie la moins contraignante, et par conséquent la moins sûre. Rien n'est bloqué et aucun message d'avertissement ne s'affiche. C'est donc la stratégie où le risque d'exécuter des scripts malveillants est le plus élevé.

Undefined : pas de stratégie d'exécution définie dans l'étendue courante. Si toutes les stratégies d'exécution de toutes les étendues sont non définies alors la stratégie effective appliquée sera la stratégie **Restricted**.

Default : positionne la stratégie par défaut, à savoir **Restricted**.

Remarque

Microsoft a mis en place ces mécanismes afin de tenter de limiter les risques liés à l'exécution de scripts provenant de l'extérieur de l'entreprise et donc potentiellement malveillants. La configuration par défaut permet d'atteindre cet objectif mais elle ne garantit en aucun cas une sécurité parfaite.

3.2.2 Les étendues des stratégies d'exécution

PowerShell permet de gérer l'étendue des stratégies. L'ordre d'application est le suivant :

- **Étendue `Process`** : la stratégie d'exécution n'affecte que la session courante (processus Windows PowerShell). La valeur affectée à l'étendue **`Process`** est stockée en mémoire uniquement ; elle n'est donc pas conservée lors de la fermeture de la session PowerShell.
- **Étendue `CurrentUser`** : la stratégie d'exécution appliquée à l'étendue **`CurrentUser`** n'affecte que l'utilisateur courant. Le type de stratégie est stocké de façon permanente dans la partie du registre **`HKEY_CURRENT_USER`**.
- **Étendue `LocalMachine`** : la stratégie d'exécution appliquée à l'étendue **`LocalMachine`** affecte tous les utilisateurs de la machine. Le type de stratégie est stocké de façon permanente dans la partie du registre **`HKEY_LOCAL_MACHINE`**.

La stratégie ayant une priorité 1 est plus propriétaire que celle ayant une priorité 3. Par conséquent, si l'étendue **`LocalMachine`** est plus restrictive que l'étendue **`Process`**, la stratégie qui s'appliquera sera quand même la stratégie de l'étendue **`Process`**. À moins que cette dernière soit de type **`Undefined`** auquel cas PowerShell appliquera la stratégie de l'étendue **`CurrentUser`** puis tentera d'appliquer la stratégie **`LocalMachine`**.

À noter que l'étendue **`LocalMachiPne`** est celle par défaut lorsque l'on applique une stratégie d'exécution sans préciser d'étendue particulière.

3.2.3 Identifier la stratégie d'exécution courante

La stratégie d'exécution courante s'obtient avec la commandelette **`Get-Execution-Policy`**.

Exemple

```
PS > Get-ExecutionPolicy
Restricted
```

Avec cette commande, nous bénéficions du commutateur **`-List`**. Grâce à lui, nous allons savoir quelles stratégies s'appliquent à nos étendues.