PROCÉDURE TECHNIQUE

Administration Windows Server avec PowerShell

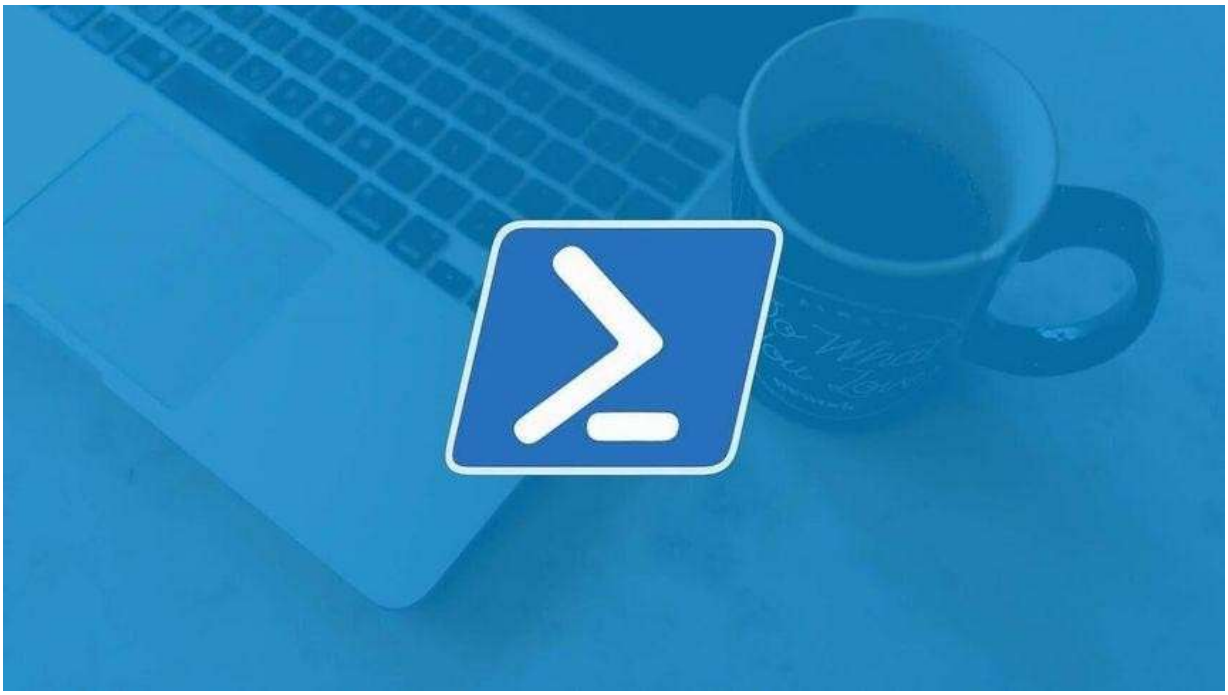

Table des matières

1. Objet et contexte

PowerShell est un langage de script et un outil d'administration permettant d'automatiser des tâches sur des serveurs Windows. Dans cette réalisation professionnelle, il est utilisé pour administrer un contrôleur de domaine Active Directory.

L'entreprise TechNova Solutions est une société spécialisée dans l'administration systèmes et réseaux. Elle agit en tant que prestataire informatique pour des entreprises tierces.

Afin d'optimiser le temps de travail des techniciens et de réduire les erreurs liées aux actions manuelles, la direction souhaite automatiser plusieurs tâches récurrentes rencontrées sur le serveur ayant le rôle de contrôleur de domaine.

Objectif : Automatiser les opérations courantes d'administration Active Directory afin de gagner du temps, d'harmoniser les pratiques et d'améliorer la sécurité du domaine.

2. Intérêt de PowerShell pour l'administration système

PowerShell permet d'administrer Windows Server en ligne de commande et par scripts. Il est particulièrement adapté aux tâches répétitives, car il permet de standardiser les actions réalisées par les techniciens.

Intérêt	Explication
Automatisation	Les tâches répétitives peuvent être exécutées plus rapidement et de manière identique.
Fiabilité	Les erreurs de saisie sont limitées grâce à des scripts qui sont préparés et testés.
Traçabilité	Les scripts peuvent écrire des logs afin de conserver une trace des actions réalisées.
Administration distante	PowerShell permet d'administrer un serveur à distance avec les droits adaptés.
Sécurité	Les comptes inactifs peuvent être détectés et désactivés automatiquement.

3. Prérequis techniques

3.1 Import du module Active Directory sur le serveur

Sur le contrôleur de domaine, le module Active Directory doit être importé avant d'utiliser les commandes de gestion AD.

```
Import-Module ActiveDirectory
```

Commande	Rôle
Import-Module ActiveDirectory	Charge les commandes PowerShell nécessaires à l'administration d'Active Directory.
Get-ADUser	Recherche ou consulte des comptes utilisateurs.

New-ADUser	Crée un nouvel utilisateur Active Directory.
Search-ADAccount	Recherche des comptes inactifs, expirés ou verrouillés.
Move-ADObject	Déplace un objet Active Directory dans une autre OU.

3.2 Installation des outils RSAT sur un poste d'administration

Si les scripts sont exécutés depuis un poste client Windows, les outils RSAT doivent être installés afin d'administrer Active Directory à distance.

```
Add-WindowsCapability -Online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
```

Prérequis d'exécution : Les scripts doivent être exécutés avec un compte disposant des droits nécessaires sur Active Directory.

4. Script 1 - Création d'un utilisateur Active Directory

4.1 Objectif du script

Ce script crée un utilisateur dans une unité d'organisation précise de l'Active Directory. Dans l'exemple, l'utilisatrice Thais ZARELLA est créée dans l'OU RH située sous l'OU Utilisateurs.

Paramètre	Valeur utilisée
Nom complet	Thais ZARELLA
Identifiant SamAccountName	t.zarella
UPN	t.zarella@ammns.local
OU de destination	OU=RH,OU=Utilisateurs,DC=AMMNS,DC=LOCAL
Mot de passe	Saisie sécurisée au moment de l'exécution

4.2 Script commenté

```
# Importation du module Active Directory.
# Il fournit les commandes New-ADUser, Get-ADUser, etc.
Import-Module ActiveDirectory


# Création du nouvel utilisateur Active Directory.
New-ADUser -Name “Thais ZARELLA” `
           -DisplayName “Thais ZARELLA” `
           -GivenName “Thais” `
           -Surname “ZARELLA” `
           -SamAccountName “t.zarella” `
           -UserPrincipalName “t.zarella@ammns.local” `
           -Path "OU=RH,OU=Utilisateurs,DC=AMMNS,DC=LOCAL" `
           -AccountPassword (Read-Host -AsSecureString "Mot de passe du nouvel utilisateur”) `
           -ChangePasswordAtLogon $true `
           -Enabled $true
```

4.3 Explication du fonctionnement

Le module Active Directory est importé au début du script.
Le compte est créé puis activé et l'utilisateur devra modifier son mot de passe à la première connexion.

5. Script 2 - Désactivation des objets inactifs

5.1 Objectif du script

Ce script recherche les comptes utilisateurs et les comptes ordinateurs inactifs depuis plus de 90 jours. Les objets trouvés sont retirés de leurs groupes secondaires, désactivés, puis déplacés dans une OU dédiée appelée Inactif.

Action	Intérêt
Rechercher les objets inactifs	Identifier les comptes qui ne sont plus utilisés.
Retirer les groupes secondaires	Réduire les droits associés aux anciens comptes.
Désactiver le compte	Empêcher toute nouvelle authentification.
Déplacer vers OU=Inactif	Isoler les comptes désactivés pour faciliter le suivi.
Ajouter une description	Conserver la date et le motif de désactivation.

5.2 Création de l'OU Inactif

Avant d'exécuter le traitement, l'unité d'organisation de destination doit exister. Le script suivant vérifie sa présence et la crée si nécessaire.

```
# Création de l'OU destinée aux objets inactifs.
New-ADOrganizationalUnit -Name "Inactif" -Path "DC=AMMNS,DC=LOCAL"
```

5.3 Script commenté

```
# Recherche des utilisateurs inactifs depuis 90 jours.
$InactivesObjects = Search-ADAccount -AccountInactive -Timespan 90 -UsersOnly |
    Where-Object { ($_.DistinguishedName -notmatch "CN=Users") -and ($_.Enabled -eq $true) }
 foreach ($Object in $InactivesObjects)
{


    $SamAccountName = $Object.SamAccountName
    $DN = $Object.DistinguishedName


    Write-Output "L'utilisateur $SamAccountName est inactif !"
    # Retirer l'utilisateur des groupes (sauf "Utilisateurs du domaine").
    Get-ADPrincipalGroupMembership -Identity $SamAccountName |
        Where-Object { $_.Name -ne "Utilisateurs du domaine" } |
        Remove-ADGroupMember -Members $SamAccountName -Confirm:$false
    # Désactiver l'utilisateur.
    Set-ADUser -Identity $SamAccountName -Enabled:$false `
        -Description "Désactivé le $(Get-Date -Format dd/MM/yyyy) pour inactivité"
    # Déplacer l'utilisateur dans l'OU Inactif.
    Move-ADObject -Identity $DN -TargetPath "OU=Inactif,DC=AMMNS,DC=LOCAL"
    Write-Output "Traitement de l'utilisateur $SamAccountName effectué."
}
```

5.4 Explication du fonctionnement

Le script recherche les objets inactifs depuis 90 jours avec Search-ADAccount.
Les groupes secondaires sont retirés afin de limiter les droits restants.
L'objet est désactivé, annoté avec une description, puis déplacé dans l'OU Inactif.

6. Script 3 - Supervision du contrôleur de domaine

6.1 Objectif du script

Ce script analyse les journaux d'événements du contrôleur de domaine afin de détecter certaines anomalies connues.

Journal	ID événement	Code
DNS Server	4013	DC-DNS-001
DFS Replication	2213	DC-DFSR-001
System	5719	DCNETLOGON-001
Directory Service	2042	DC-AD-001

6.2 Script commenté

```
# Contrôleur(s) de domaine à superviser.
$DCs = @("AMMNS.local")


# Règles de supervision : journal, ID, code et explication.
$Rules = @(
    @{ Log = "DNS Server";        Id = 4013; Code = "DC-DNS-001";      Resume = "Le service DNS ne s'initialise pas correctement." },
    @{ Log = "DFS Replication";   Id = 2213; Code = "DC-DFSR-001";     Resume = "Le service DFSR est bloqué après un arrêt non propre." },
    @{ Log = "System";            Id = 5719; Code = "DC-NETLOGON-001"; Resume = "Netlogon rencontre une anomalie de communication." },
    @{ Log = "Directory Service"; Id = 2042; Code = "DC-AD-001";       Resume = "La réplication Active Directory est bloquée." }
)


# Parcours de chaque contrôleur et de chaque
règle. foreach ($DC in $DCs) {     foreach ($Rule
in $Rules) {
        # Recherche de TOUS les événements correspondant à la règle (sans limite de temps).
        $Events = Get-WinEvent -ComputerName $DC -FilterHashtable @{
            LogName = $Rule.Log
            Id      = $Rule.Id
        } -ErrorAction SilentlyContinue


        # Affichage du code d'erreur personnalisé et de son explication.
        foreach ($Event in $Events) {
            Write-Host "$($Event.TimeCreated) | $DC | $($Rule.Code) | $($Rule.Resume) | EventID=$($Event.Id)"
}
    }
}
```

6.3 Explication du fonctionnement

Le script analyse les journaux System, Directory Service, DNS Server et DFS Replication.

Lorsqu'il détecte une erreur, il le résume en donnant un code d'erreur personnalisé en expliquant le problème.

7. Tests et vérifications

Test	Commande / action	Résultat
Module AD	Get-Module ActiveDirectory	Le module ActiveDirectory est chargé.
Création utilisateur	Vérifier dans l'OU RH	Le compte t.zarella est présent et activé.
Objets inactifs	Exécuter le script en test	Les objets inactifs sont déplacés vers OU=Inactif.
Services supervisés	Get-Service DNS, DFSR, Netlogon	Les services sont dans un état cohérent.

8. Bonnes pratiques de sécurité

Ne jamais stocker un mot de passe en clair dans un script PowerShell.
Exécuter les scripts avec un compte disposant uniquement des droits nécessaires.
Tester les scripts dans une maquette avant une utilisation en production.
Conserver des logs pour tracer les actions automatisées.
Limiter les redémarrages automatiques à des services précisément identifiés.
Documenter les OU, les comptes de service et les chemins utilisés. Sauvegarder l'Active Directory avant une modification de masse.

9. Récapitulatif des commandes utiles

Objectif	Commande
Importer le module AD	Import-Module ActiveDirectory
Installer RSAT	Add-WindowsCapability -Online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
Créer un utilisateur	New-ADUser
Rechercher les comptes inactifs	Search-ADAccount -AccountInactive
Désactiver un utilisateur	Set-ADUser -Enabled $false
Désactiver un ordinateur	Set-ADComputer -Enabled $false
Déplacer un objet AD	Move-ADObject
Lire les journaux Windows	Get-WinEvent
Redémarrer un service	Restart-Service

10. Conclusion

Les scripts PowerShell présentés dans cette procédure permettent d'automatiser plusieurs tâches importantes d'administration d'un contrôleur de domaine Windows Server. Ils répondent à des besoins

concrets : création de comptes utilisateurs, traitement des objets inactifs et supervision des services liés au domaine.

Cette automatisation permet à TechNova Solutions de gagner du temps, de standardiser ses interventions et de renforcer la sécurité de l'Active Directory en réduisant les comptes inutilisés et en améliorant la traçabilité des incidents.