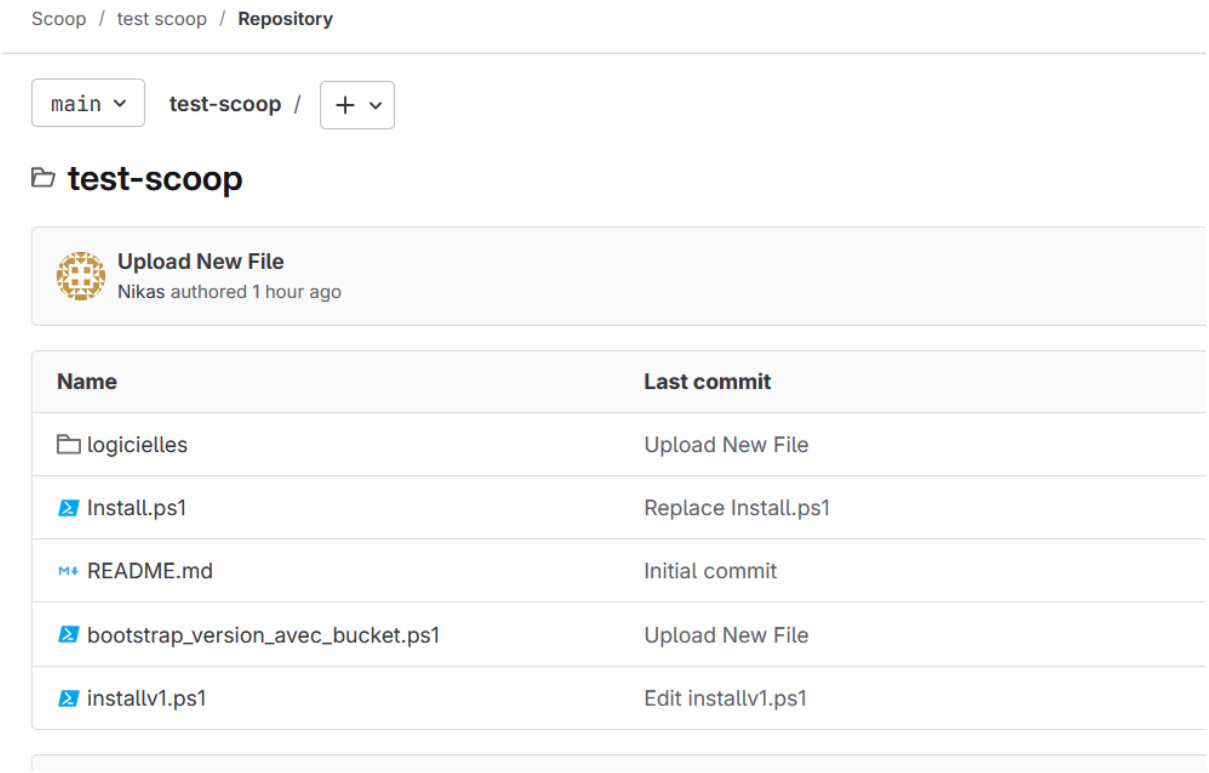
WINDOWS 10

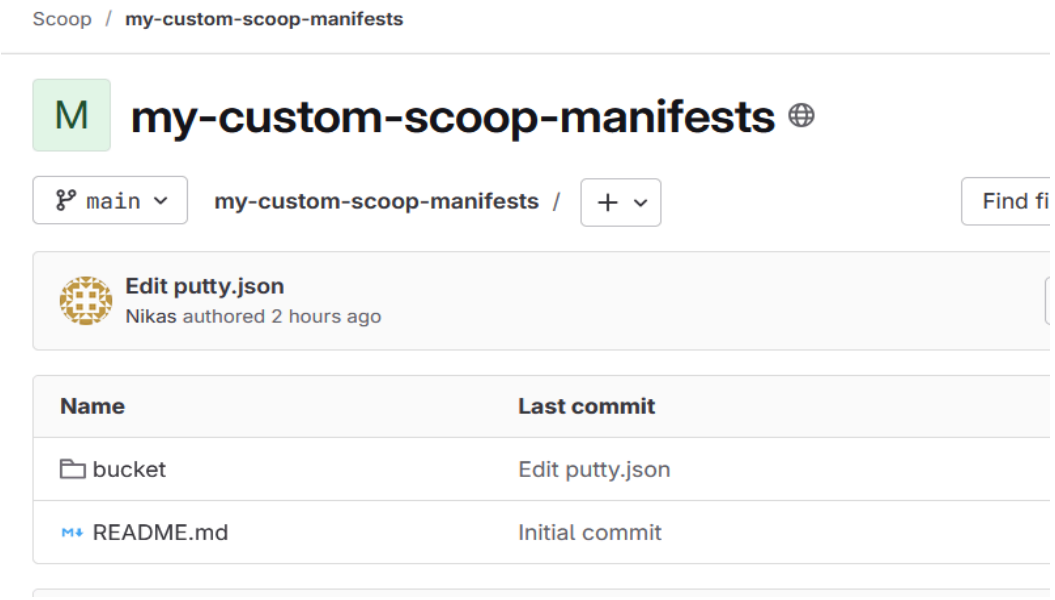
Étape 1: Créer les 2 dépôts nécessaires à la réalisation du projet (buckets gérant les manifests et un contenant les scripts principal et les installateurs).

Scoop est un **gestionnaire de paquets pour Windows** basé sur PowerShell. Il permet d'installer facilement des logiciels via des **scripts d'installation automatisés (manifests)**.

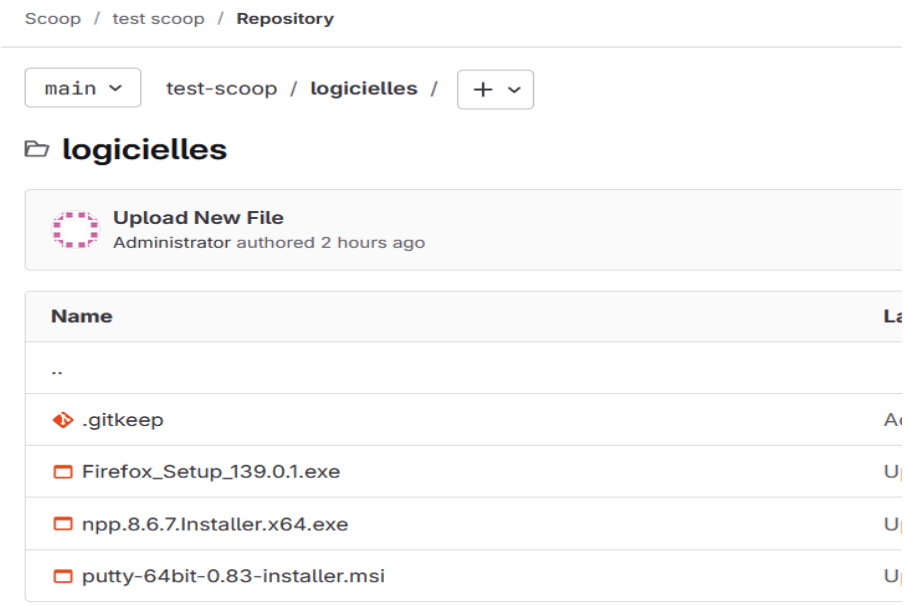
Dépôts test-scoop



Dépôts des manifests

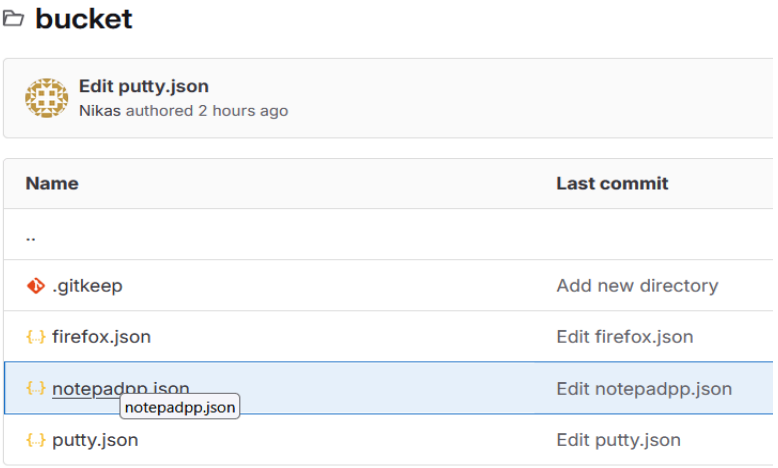


Étape 2 : Ajouter les installeurs dans un dossier (télécharger les fichiers puis les importer un à un).



Étape 3 : Créer un buckets qui contient les manifests de chaque logiciel.
Un bucket est une collection de manifests. Un manifest est un fichier .json utilisé par Scoop pour décrire comment installer un logiciel.
Il contient :

- version : numéro de version du logiciel
- url : lien direct vers l'installeur (ou le fichier portable)
- bin : nom du fichier exécutable à rendre accessible
- installer : (optionnel) script pour les .msi, .exe, etc.
- shortcuts : (optionnel) raccourcis à créer
- hash : empreinte SHA256 du fichier (vérifie l'intégrité)



Exemple d’un manifest :

notepadpp.json 756 B

```json
{
  "version": "8.6.7",
  "description": "Notepad++ éditeur de texte avancé",
  "homepage": "https://notepad-plus-plus.org",
  "license": "GPL-3.0",
  "architecture": {
    "64bit": {
      "url": "https://[redacted]/scoop/test-scoop/-/raw/main/logicielles/npp.8.6.7.Installer.x64.exe",
      "hash": "81470EB5917705FA0DF03181B81124226718 42BDCEC5252A7894975B38058C91"
    }
  },
  "installer": {
    "script": [
      "Start-Process -Wait \"$dir\\npp.8.6.7.Installer.x64.exe\" '/S'",
      "Copy-Item \"$env:ProgramFiles\\Notepad++\\notepad++.exe\" \"$dir\\notepad++.exe\" -ErrorAction SilentlyContinue"
    ]
  },
  "bin": [
    "notepad++.exe"
  ],
  "shortcuts": [
    [
      "notepad++.exe",
      "Notepad++"
    ]
  ]
}
```

Explication de la partie script de firefox (problème rencontré: le shim n'était pas crée peut-être à cause du fichiers de base donc on a utiliser un nouveau mais par mesure de sécurité le script a été adapté pour ça) :

```
"installer": {
  "script": [
    "Start-Process -FilePath \"$dir\\Firefox_Setup_139.0.1.exe\" -ArgumentList '/S' -Wait",
    "# Copie du vrai firefox.exe installé dans Program Files vers le dossier actuel",
    "$firefoxPath = 'C:\\Program Files\\Mozilla Firefox\\firefox.exe'",
    "if (Test-Path $firefoxPath) { Copy-Item $firefoxPath -Destination \"$dir\\firefox.exe\" -Force }"
  ]
},
```

// Exécution silencieuse de l’installeur .exe

"Start-Process -FilePath \"$dir\\Firefox_Setup_139.0.1.exe\" -ArgumentList '/S' -Wait",

// Message visible pendant l'installation

"Write-Host 'Copie du vrai firefox.exe installé dans Program Files vers le dossier actuel'",

// Chemin du binaire Firefox

"$firefoxPath = 'C:\\Program Files\\Mozilla Firefox\\firefox.exe'",

// Si le binaire existe, on le copie vers le dossier Scoop

"if (Test-Path $firefoxPath) { Copy-Item $firefoxPath -Destination \"$dir\\firefox.exe\" -Force }"

Explication de la partie script de notepad++ (le plus simple 0 problème rencontré):

```
"installer": {
  "script": [
    "Start-Process -Wait \"$dir\\npp.8.6.7.Installer.x64.exe\" '/S'",
    "Copy-Item \"$env:ProgramFiles\\Notepad++\\notepad++.exe\" \"$dir\\notepad++.exe\" -ErrorAction SilentlyContinue"
  ]
},
```

// Lance l'installeur en mode silencieux (paramètre '/S') et attend la fin de l'installation

"Start-Process -Wait \"$dir\\npp.8.6.7.Installer.x64.exe\" '/S'",

// Copie le binaire installé dans Program Files vers le dossier d'installation de Scoop

// Cela permet à Scoop de créer un "shim" (lien exécutable) vers le vrai fichier

// SilentlyContinue Empêche d'afficher une erreur si le fichier n'existe pas

"Copy-Item \"$env:ProgramFiles\\Notepad++\\notepad++.exe\" \"$dir\\notepad++.exe\" -ErrorAction SilentlyContinue"

Explication de la partie script de putty (problème rencontré : le shim n'était pas à la destination attendu et c'est un .msi donc plusieurs fichiers .exe donc on a du adapter le script aussi):

```
"installer": {
  "script": [
    "$sourceDir = \"$dir\\PFiles\\PuTTY\"",
    "Get-ChildItem $sourceDir -Filter *.exe | ForEach-Object { Copy-Item $_.FullName -Destination $dir }"
  ]
},
```

// Définition du chemin vers le dossier où les exécutables ont été extraits par l'installeur MSI

// Ici, 'PFiles\\PuTTY' correspond à "Program Files\\PuTTY" recréé dans le dossier temporaire d'installation de Scoop

"$sourceDir = \"$dir\\PFiles\\PuTTY\"",

// Parcours tous les fichiers .exe dans ce dossier source

// Pour chaque fichier, copie-le dans le dossier principal $dir (celui où Scoop installe le logiciel)

// Cela permet ensuite à Scoop de créer des shims fonctionnels pour chaque exécutable

"Get-ChildItem $sourceDir -Filter *.exe | ForEach-Object { Copy-Item $_.FullName -Destination $dir }"

Un shim est un petit exécutable intermédiaire créé par Scoop dans son dossier shims. Le shim permet d'exécuter un programme depuis n'importe où dans le terminal, même s'il est installé ailleurs.

Étape 4 : Créer le script bootstrap.

C’est un script de démarrage qui :

1. Télécharge un autre script depuis GitLab
2. L'exécute automatiquement
3. Contourne les problèmes de certificat SSL si besoin

C’est utile pour lancer facilement une installation depuis Internet (ou un GitLab privé).

Script commenté pour comprendre chaque étape principale du script :

bootstrap.ps1 2.71 KiB

```powershell
# SCRIPT BOOTSTRAP D'INSTALLATION ADAPTÉ POUR PROJETS PUBLICS EN HTTPS
# Rôle : Télécharger et exécuter le script principal 'Install.ps1' depuis GitLab.

# Début des contournements SSL/TLS (nécessaires si le certificat n'est pas reconnu)
Add-Type -AssemblyName System.Net.Http  # Chargement du type nécessaire pour gérer les requêtes HTTP
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -bor [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolT
[Net.ServicePointManager]::ServerCertificateValidationCallback = { $true }  # Contournement de la validation des certificats SSL (NON RECOMMANDÉ e
# Fin des contournements SSL/TLS

# Les projets GitLab étant publics, aucun token d'accès n'est requis sinon c'est le cas.

# CONFIGURATION DE BASE
$gitlabGroup = "Scoop"          # Nom du groupe GitLab où se trouve le projet
$gitlabRepo = "test-scoop"      # Nom du dépôt GitLab contenant le script
$branch = "main"                # Nom de la branche du projet à utiliser

# URL directe pour télécharger le script principal 'Install.ps1' depuis GitLab
$scriptDownloadUrl = "https://[redacted]$gitlabGroup/$gitlabRepo/-/raw/$branch/Install.ps1?ref_type=heads"

# Chemin temporaire où le script principal sera téléchargé localement
$tempScriptPath = Join-Path $env:TEMP "install_downloaded_from_gitlab.ps1"

# TÉLÉCHARGEMENT ET EXÉCUTION
Write-Host "`n🌐 Tentative de téléchargement du script principal (Install.ps1) depuis GitLab..."
Write-Host "URL de téléchargement tentée : $scriptDownloadUrl"

try {
    # Téléchargement du script principal depuis GitLab
    Invoke-WebRequest -Uri $scriptDownloadUrl -OutFile $tempScriptPath -UseBasicParsing
    Write-Host "✅ Script principal téléchargé avec succès dans : $tempScriptPath"

    # Exécution du script principal téléchargé
    Write-Host "🚀 Exécution du script principal..."
    & $tempScriptPath
    Write-Host "🏁 Exécution du script principal terminée."
}
catch {
    Write-Error "❌ Erreur critique lors du téléchargement ou de l'exécution du script principal."
    Write-Error $_.Exception.Message
    Write-Error "URL complète tentée : $scriptDownloadUrl"
    Write-Error "Vérifiez :"
    Write-Error "1. La connectivité HTTPS vers [redacted]"
    Write-Error "2. Le nom et les chemins EXACTS du fichier Install.ps1 dans votre dépôt GitLab (Public)"
    Write-Error "3. Que les contournements SSL/TLS sont bien actifs si votre certificat n'est pas reconnu"
}

# Nettoyage (facultatif désactivé en version de test) : Supprime le script téléchargé après son exécution
# Remove-Item $tempScriptPath -ErrorAction SilentlyContinue
# Write-Host "🧹 Script temporaire supprimé."
```

Étape 5 : Créer le script principal Install.ps1

Il :

- Installe Scoop si besoin
- Ajoute ton bucket GitLab personnalisé
- Installe les logiciels listés

Il est téléchargé automatiquement via `bootstrap`.

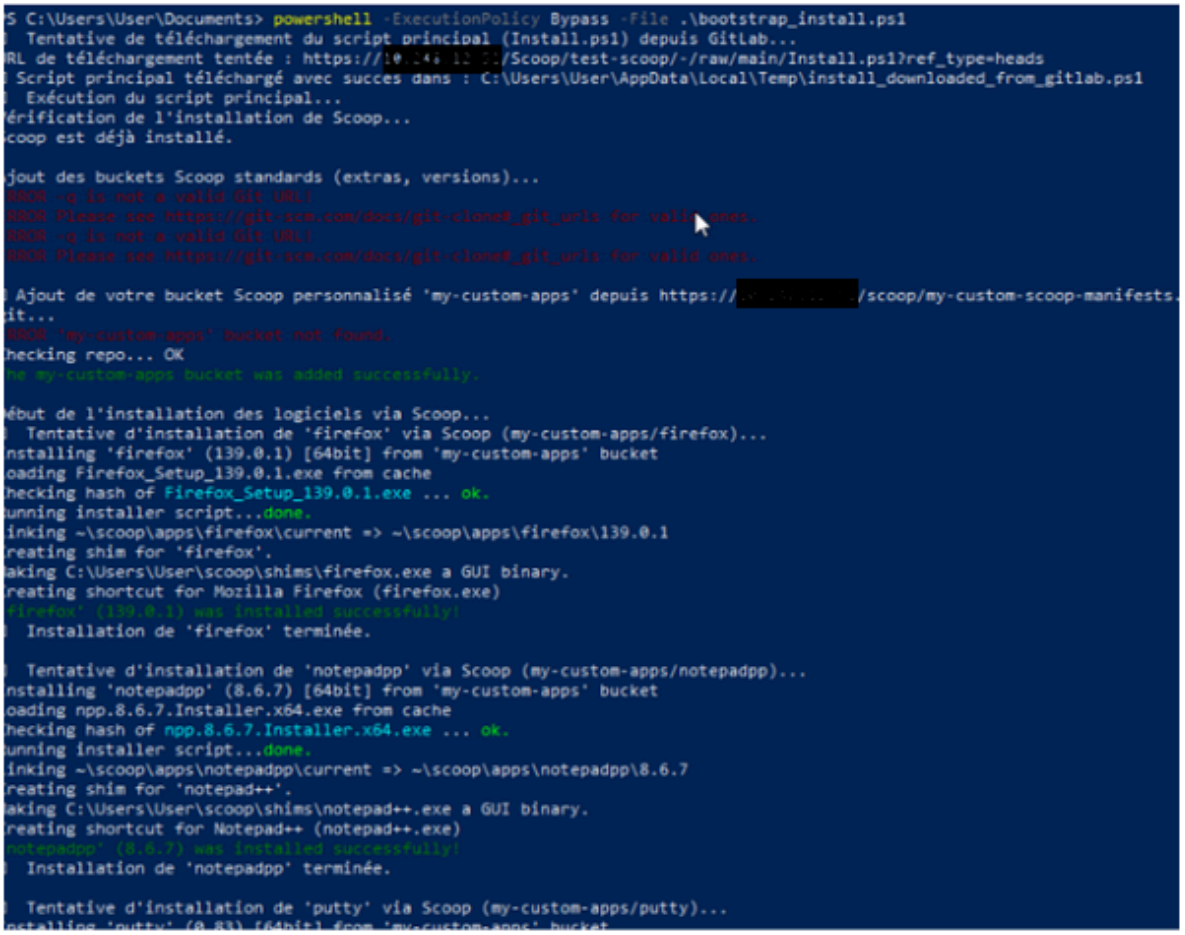
Install.ps1 3.55 KiB

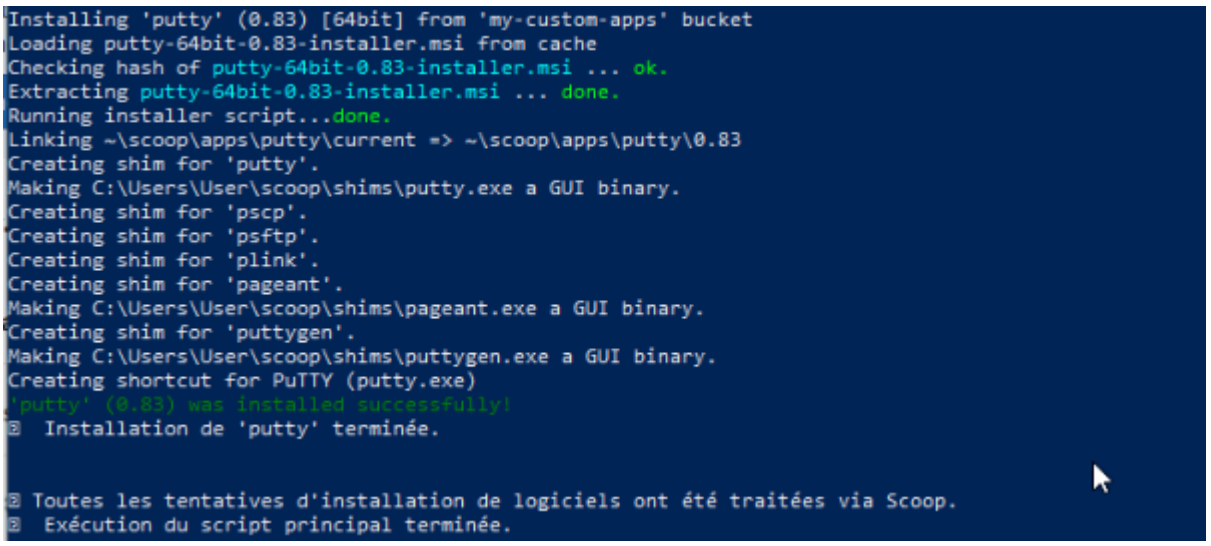
```powershell
# SCRIPT D'INSTALLATION DE LOGICIELS

# Début des contournements SSL/TLS nécessaires pour les certificats non reconnus
Add-Type -AssemblyName System.Net.Http
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -bor [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolT
[Net.ServicePointManager]::ServerCertificateValidationCallback = { $true }
# Fin des contournements SSL/TLS

# CONFIGURATION DE BASE

$gitlabGroup             = "Scoop"  # Nom du groupe GitLab pour les dépôts Scoop
$gitlabRepoForManifests  = "my-custom-scoop-manifests"  # Nom du dépôt GitLab contenant les manifestes Scoop personnalisés
$gitlabRepoForExecutables = "test-scoop"  # Nom du dépôt GitLab contenant les exécutables
$branch                  = "main"  # Branche à utiliser pour les dépôts GitLab

$customBucketName = "my-custom-apps"  # Nom du bucket Scoop personnalisé
$customBucketUrl  = "https://[redacted]/scoop/my-custom-scoop-manifests.git"  # URL du dépôt GitLab pour les manifestes Scoop personnalisés
$executablesBaseUrl = "https://[redacted]/scoop/test-scoop/-/raw/$branch"  # URL de base pour les exécutables

# INSTALLATION DE SCOOP

Write-Host "Vérification de l'installation de Scoop..."
if (-not (Test-Path "$env:USERPROFILE\scoop")) {  # Vérification si Scoop est installé
    Write-Host "🔧 Installation de Scoop..."
    Set-ExecutionPolicy RemoteSigned -Scope Process -Force  # Définition de la politique d'exécution pour autoriser l'installation
    Invoke-RestMethod get.scoop.sh | Invoke-Expression  # Téléchargement et exécution du script d'installation de Scoop
    Write-Host "Scoop devrait être installé. Redémarrage de la session PowerShell recommandé si des problèmes surviennent."
} else {
    Write-Host "Scoop est déjà installé."
}

# AJOUT DES BUCKETS

Write-Host "`nAjout des buckets Scoop standards (extras, versions)..."
scoop bucket add extras -q 2>$null  # Ajout du bucket 'extras' pour les logiciels standards
scoop bucket add versions -q 2>$null  # Ajout du bucket 'versions' pour les anciennes versions des logiciels

Write-Host "`n➕ Ajout de votre bucket Scoop personnalisé '$customBucketName' depuis $customBucketUrl..."
scoop bucket rm $customBucketName -q 2>$null  # Suppression de l'ancien bucket personnalisé s'il existe déjà
scoop bucket add $customBucketName $customBucketUrl  # Ajout du bucket personnalisé depuis l'URL GitLab

# INSTALLATION DES LOGICIELS

$logicielsAInstaller = @(
    "firefox",  # Liste des logiciels à installer via Scoop
    "notepadpp",
    "putty"
)

Write-Host "`nDébut de l'installation des logiciels via Scoop..."
foreach ($nomApp in $logicielsAInstaller) {  # Parcours de la liste des logiciels à installer
    $nomComplet = "$customBucketName/$nomApp"  # Création du nom complet pour le bucket et le logiciel
    Write-Host "📦 Tentative d'installation de '$nomApp' via Scoop ($nomComplet)..."
    try {
        scoop install $nomComplet  # Tentative d'installation du logiciel via Scoop
        Write-Host "🎉 Installation de '$nomApp' terminée.`n"
    } catch {
        Write-Host "❌ Erreur lors de l'installation de '$nomApp' via Scoop."
        Write-Host $_.Exception.Message
        Write-Host "Vérifiez :"
        Write-Host "1. La présence du manifeste '$nomApp.json' dans le dossier 'bucket/' de '$gitlabRepoForManifests'."
        Write-Host "2. Que l'URL DANS le manifeste pointe bien vers '$executablesBaseUrl/logicielles/${nomApp}_Installer.exe'.`n"
    }
}

Write-Host "`n✅ Toutes les tentatives d'installation de logiciels ont été traitées via Scoop."
```

De préférence il faut faire toutes les modifications de script que ça soit bootstrap ou Install sur powershell ISE comme ça on peut être sûr que le script est au bon format UTF-8 et éviter toutes les erreurs qu'on a rencontré à cause d'un mauvais formatage.

Etape 6 : TEST sur la VM windows 10 et 11

1. lancer powershell en tant qu'administrateur
2. se positionner dans le bon dossier si le script bootstrap a été téléchargé avec cd sinon télécharger manuellement ou avec Invoke-WebRequest -Uri "https://10.248.12.51/Scoop/test-scoop/-/raw/main/bootstrap.ps1" -OutFile "$env:TEMP\bootstrap.ps1" -UseBasicParsing + powershell -ExecutionPolicy Bypass -File "$env:TEMP\bootstrap.ps1"
3. Où faire la commande powershell -ExecutionPolicy Bypass -File .\bootstrap.ps1





WINDOWS 11

Différence majeurs de Install2.ps1 par rapport au premier :

- Gestion des erreurs : Complète avec try/catch à chaque étape
- Compatibilité stricte : Plus défensive, vérifie tout étape par étape

- Logs : Messages explicites sur les causes possibles d’échecs

Install2.ps1 3.21 KiB

```powershell
# INSTALL2.PS1 – Installation de logiciels via Scoop avec contournement SSL (Windows 11)

# Chargement de HttpClient si nécessaire
Add-Type -AssemblyName System.Net.Http

# Contournement SSL/TLS pour certificats non reconnus
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -bor [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolT
[Net.ServicePointManager]::ServerCertificateValidationCallback = { $true }

# CONFIGURATION DE BASE

$gitlabGroup              = "Scoop"
$gitlabRepoForManifests   = "my-custom-scoop-manifests"
$gitlabRepoForExecutables = "test-scoop"
$branch                   = "main"

$customBucketName         = "my-custom-apps"
$customBucketUrl          = "https://[redacted]/scoop/$gitlabRepoForManifests.git"
$executablesBaseUrl       = "https://[redacted]scoop/$gitlabRepoForExecutables/-/raw/$branch"

# Liste des apps à installer (associée à leur nom de manifeste et exécutable attendu)

$logicielsAInstaller = @(
    "firefox",
    "notepadpp",
    "putty"
)

# INSTALLATION DE SCOOP

Write-Host "`n🔍 Vérification de Scoop..."
if (-not (Test-Path "$env:USERPROFILE\scoop")) {
    try {
        Write-Host "🛠️ Scoop non détecté. Installation en cours..."
        Set-ExecutionPolicy RemoteSigned -Scope Process -Force
        Invoke-RestMethod get.scoop.sh | Invoke-Expression
        Write-Host "✅ Scoop installé avec succès."
    } catch {
        Write-Error "❌ Impossible d'installer Scoop."
        Write-Error $_.Exception.Message
        exit 1
    }
} else {
    Write-Host "✅ Scoop est déjà installé."
}

# AJOUT DES BUCKETS

Write-Host "`n📦 Ajout des buckets standards..."
foreach ($stdBucket in @("extras", "versions")) {
    try {
        scoop bucket add $stdBucket -q 2>$null
        Write-Host "✅ Bucket ajouté : $stdBucket"
    } catch {
        Write-Host "⚠️ Bucket déjà présent : $stdBucket"
    }
```

```powershell

Write-Host "`n📁 Ajout du bucket personnalisé '$customBucketName'..."
try {
    scoop bucket rm $customBucketName -q 2>$null
} catch {
    Write-Host "ℹ️ Suppression silencieuse du bucket existant (s'il existait)."
}
try {
    scoop bucket add $customBucketName $customBucketUrl
    Write-Host "✅ Bucket personnalisé ajouté avec succès."
} catch {
    Write-Error "❌ Échec lors de l'ajout du bucket personnalisé : $customBucketUrl"
    Write-Error $_.Exception.Message
    exit 1
}

# INSTALLATION DES LOGICIELS
Write-Host "`n🚀 Lancement de l'installation des logiciels..."

foreach ($app in $logicielsAInstaller) {
    $manifestPath = "$customBucketName/$app"
    Write-Host "`n→ Installation de '$app' via $manifestPath"
    try {
        scoop install $manifestPath
        Write-Host "✅ $app installé avec succès."
    } catch {
        Write-Warning "❌ Échec de l'installation de '$app'."
        Write-Host "🩹 Étapes de vérification recommandées :"
        Write-Host " - 🔍 Vérifiez que '$app.json' existe dans le dépôt '$gitlabRepoForManifests/bucket/'"
        Write-Host " - 📁 Chemin attendu vers l'exécutable :"
        Write-Host "   $executablesBaseUrl/logicielles/${app}_Installer.exe"
    }
}

Write-Host "`n🏁 Script terminé. Toutes les installations ont été traitées."
```

Différence majeurs de Install2.ps1 par rapport au premier :

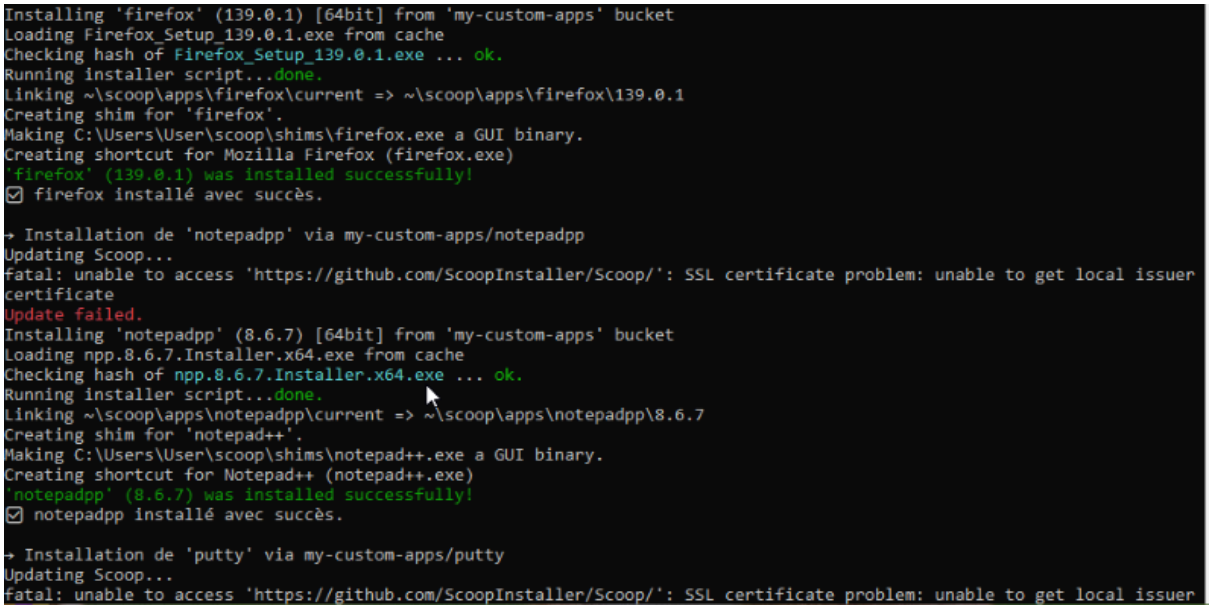
- Contournement SSL : TrustAllCertsPolicy (ancienne méthode .NET)

bootstrap2.ps1 1.59 KiB

```powershell
# SCRIPT BOOTSTRAP2 Windows 11
# Rôle : Télécharge et exécute le script principal 'Install2.ps1' depuis GitLab, en contournant les erreurs SSL.

# Variables GitLab
$gitlabGroup = "Scoop"
$gitlabRepo = "test-scoop"
$branch = "main"
$scriptDownloadUrl = "https://[redacted]/$gitlabGroup/$gitlabRepo/-/raw/$branch/Install2.ps1?ref_type=heads"
$tempScriptPath = Join-Path $env:TEMP "install_downloaded_from_gitlab.ps1"

Write-Host "🌐 Téléchargement du script principal (Install2.ps1) depuis GitLab via Invoke-WebRequest..."
Write-Host "URL : $scriptDownloadUrl"

try {
    # Contournement SSL (nécessite que la version de .NET le permette)
    add-type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
    public bool CheckValidationResult(
        ServicePoint srvPoint, X509Certificate certificate,
        WebRequest request, int certificateProblem) {
        return true;
    }
}
"@
    [System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy

    Invoke-WebRequest -Uri $scriptDownloadUrl -OutFile $tempScriptPath -UseBasicParsing

    Write-Host "✅ Script téléchargé avec succès dans : $tempScriptPath"
    Write-Host "🚀 Exécution du script..."

    & $tempScriptPath

    Write-Host "🎉 Exécution du script principale terminée."
}
catch {
    Write-Error "❌ Échec du téléchargement ou de l'exécution du script principal."
    Write-Error $_.Exception.Message
    Write-Error "Vérifiez que l'URL est correcte, que GitLab est accessible et que le certificat est contourné."
}
```

TEST :


PS C:\Users\User\Downloads> powershell -ExecutionPolicy Bypass -File .\bootstrap2.ps1
Téléchargement du script principal (Install2.ps1) depuis GitLab via Invoke-WebRequest...
URL : https://10.248.12.51/Scoop/test-scoop/-/raw/main/Install2.ps1?ref_type=heads
Script téléchargé avec succès dans : C:\Users\User\AppData\Local\Temp\install_downloaded_from_gitlab.ps1
Exécution du script...
Vérification de Scoop...
Scoop est déjà installé.
Ajout des buckets standards...
ERROR -q is not a valid Git URL!
ERROR Please see https://git-scm.com/docs/git-clone#_git_urls for valid ones.
Bucket ajouté : extras
ERROR -q is not a valid Git URL!
ERROR Please see https://git-scm.com/docs/git-clone#_git_urls for valid ones.
Bucket ajouté : versions
Ajout du bucket personnalisé 'my-custom-apps'...
The my-custom-apps bucket was removed successfully.
Checking repo... OK
The my-custom-apps bucket was added successfully.
Bucket personnalisé ajouté avec succès.
Lancement de l'installation des logiciels...
→ Installation de 'firefox' via my-custom-apps/firefox
Updating Scoop...
fatal: unable to access 'https://github.com/ScoopInstaller/Scoop/': SSL certificate problem: unable to get local issuer certificate
Update failed.


Installing 'firefox' (139.0.1) [64bit] from 'my-custom-apps' bucket
Loading Firefox_Setup_139.0.1.exe from cache
Checking hash of Firefox_Setup_139.0.1.exe ... ok.
Running installer script...done.
Linking ~\scoop\apps\firefox\current => ~\scoop\apps\firefox\139.0.1
Creating shim for 'firefox'.
Making C:\Users\User\scoop\shims\firefox.exe a GUI binary.
Creating shortcut for Mozilla Firefox (firefox.exe)
'firefox' (139.0.1) was installed successfully!
firefox installé avec succès.
→ Installation de 'notepadpp' via my-custom-apps/notepadpp
Updating Scoop...
fatal: unable to access 'https://github.com/ScoopInstaller/Scoop/': SSL certificate problem: unable to get local issuer certificate
Update failed.
Installing 'notepadpp' (8.6.7) [64bit] from 'my-custom-apps' bucket
Loading npp.8.6.7.Installer.x64.exe from cache
Checking hash of npp.8.6.7.Installer.x64.exe ... ok.
Running installer script...done.
Linking ~\scoop\apps\notepadpp\current => ~\scoop\apps\notepadpp\8.6.7
Creating shim for 'notepad++'.
Making C:\Users\User\scoop\shims\notepad++.exe a GUI binary.
Creating shortcut for Notepad++ (notepad++.exe)
'notepadpp' (8.6.7) was installed successfully!
notepadpp installé avec succès.
→ Installation de 'putty' via my-custom-apps/putty
Updating Scoop...
fatal: unable to access 'https://github.com/ScoopInstaller/Scoop/': SSL certificate problem: unable to get local issuer

```
certificate
Update failed.
Installing 'putty' (0.83) [64bit] from 'my-custom-apps' bucket
Loading putty-64bit-0.83-installer.msi from cache
Checking hash of putty-64bit-0.83-installer.msi ... ok.
Extracting putty-64bit-0.83-installer.msi ... done.
Running installer script...done.
Linking ~\scoop\apps\putty\current => ~\scoop\apps\putty\0.83
Creating shim for 'putty'.
Making C:\Users\User\scoop\shims\putty.exe a GUI binary.
Creating shim for 'pscp'.
Creating shim for 'psftp'.
Creating shim for 'plink'.
Creating shim for 'pageant'.
Making C:\Users\User\scoop\shims\pageant.exe a GUI binary.
Creating shim for 'puttygen'.
Making C:\Users\User\scoop\shims\puttygen.exe a GUI binary.
Creating shortcut for PuTTY (putty.exe)
'putty' (0.83) was installed successfully!
☑ putty installé avec succès.

▩ Script terminé. Toutes les installations ont été traitées.
Exécution du script principal terminée.
```

Remarque : bootstrap2.ps1 et Install2.ps1 fonctionne aussi sur windows 10, donc c'est le script à privilégier.

Seconde remarque après 10 min : après exécution de bootstrap erreur j'ai pas fouiller pour en avoir la raison puisque bootstrap2 et Install2 fonctionne toujours. Mais bootstrap2.ps1 fonctionne même s'il pointe vers Install.ps1 donc le problème vient de bootstrap, surement un problème lié au contournement SSL.

```
PS C:\Windows\system32> cd C:\Users\User\Downloads\
PS C:\Users\User\Downloads> powershell -ExecutionPolicy ByPass -File .\bootstrap.ps1

  Tentative de téléchargement du script principal (Install.ps1) depuis GitLab...
URL de téléchargement tentée : https://10.[redacted]/Scoop/test-scoop/-/raw/main/Install.ps1?ref_type=heads
C:\Users\User\Downloads\bootstrap.ps1 : ⊠ Erreur critique lors du téléchargement ou de l'exécution du script principal.
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : La connexion sous-jacente a été fermée: Une erreur inattendue s'est produite
lors de l'envoi.
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : URL complète tentée :
https://[redacted]/Scoop/test-scoop/-/raw/main/Install.ps1?ref_type=heads
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : Vérifiez :
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : 1. La connectivité HTTPS vers [redacted]
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : 2. Le nom et les chemins EXACTS du fichier Install.ps1 dans votre dépôt GitLab
(Public)
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1

C:\Users\User\Downloads\bootstrap.ps1 : 3. Que les contournements SSL/TLS sont bien actifs si votre certificat n'est
pas reconnu
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,bootstrap.ps1
```