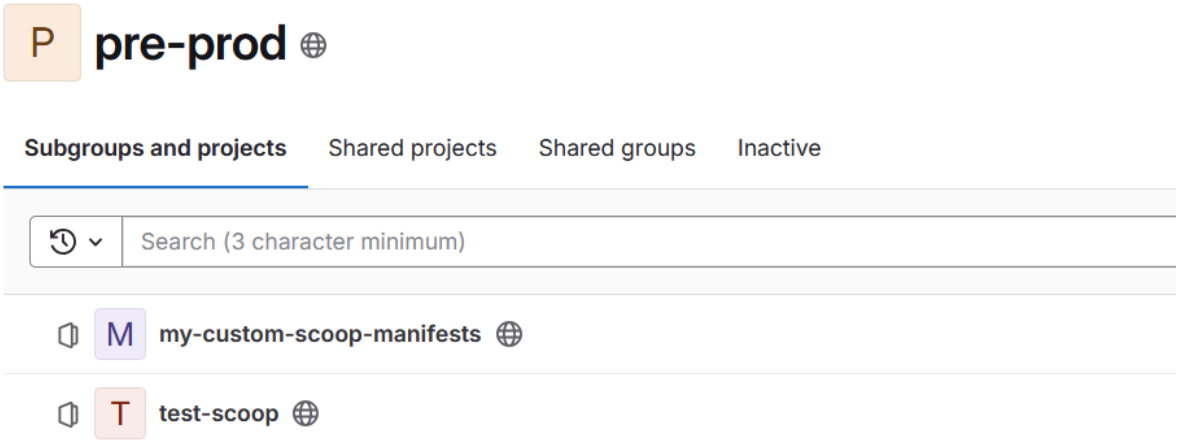
Scoop annexe

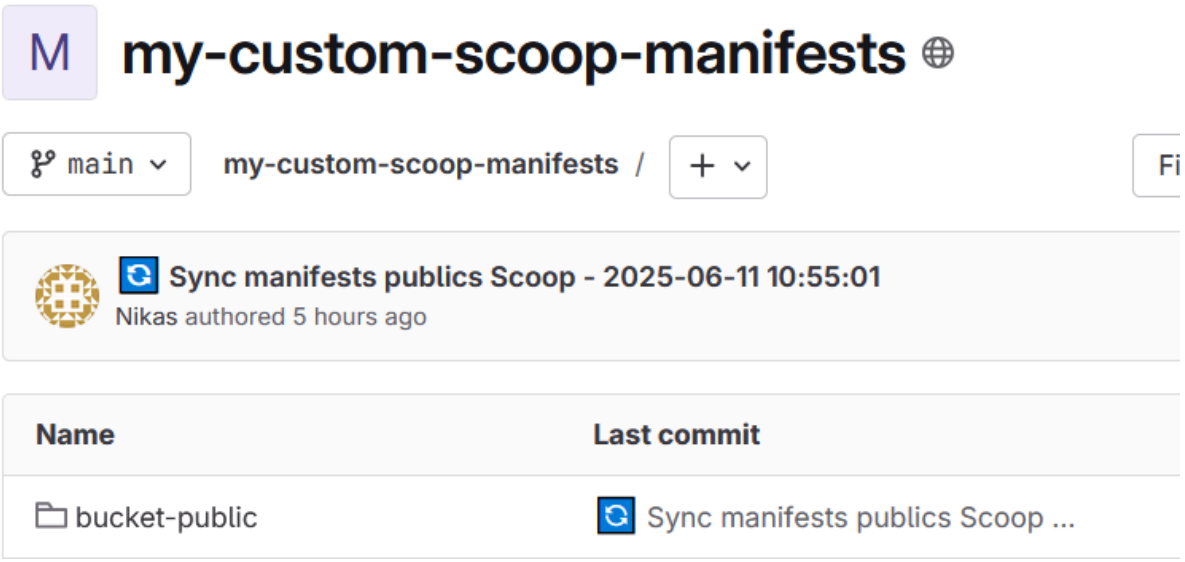
Créer un environnement de pré-production pour tester les nouvelles versions logicielles sur une dizaine de machines avant leur déploiement généralisé.




Ce nouveau groupe contient deux dépôts, comme le précédent. Il permettra de tester les nouvelles versions des logiciels et intègre également des utilités supplémentaires, qui seront développées dans les missions suivantes. Chaque dépôt reprend la même structure que ceux du premier groupe.

Tenter la synchronisation des manifestes du bucket public (Scoop) avec notre bucket privé.

Synchroniser les buckets Scoop publics permet de disposer de nombreux manifestes pour différents logiciels, tout en conservant un contrôle total sur leur contenu et leur intégrité. Ils sont stockés dans l'environnement de pré-production afin que les tests puissent être réalisés avec un risque limité.




la synchronisation c'est faites à travers un script powershell :

sync-public-buckets.ps1 2.06 KiB

```powershell
# CONFIGURATION
$bucketSources = @(
    "https://github.com/ScoopInstaller/Main",
    "https://github.com/ScoopInstaller/Extras",
    "https://github.com/ScoopInstaller/Versions"
)

$tempDir = "$env:TEMP\scoop-public-sync"
$localRepoDir = "C:\Users\User\my-custom-scoop-manifests"
$publicBucketDir = Join-Path $localRepoDir "bucket-public"

# Nettoyage temporaire
Remove-Item -Recurse -Force $tempDir -ErrorAction SilentlyContinue
New-Item -ItemType Directory -Path $tempDir -Force | Out-Null
New-Item -ItemType Directory -Path $publicBucketDir -Force | Out-Null

# CLONAGE + COPIE DES MANIFESTES JSON
foreach ($source in $bucketSources) {
    $repoName = ($source.Split("/")[-1])
    $clonePath = Join-Path $tempDir $repoName

    Write-Host "`n--- Clonage de $repoName ---"
    git clone --depth=1 $source $clonePath

    $jsonFiles = Get-ChildItem -Path $clonePath -Recurse -Include *.json

    foreach ($file in $jsonFiles) {
        $dest = Join-Path $publicBucketDir $file.Name

        $shouldCopy = $true

        if (Test-Path $dest) {
            # Compare le contenu des fichiers (ligne à ligne)
            $sourceContent = Get-Content $file.FullName -Raw
            $destContent = Get-Content $dest -Raw

            if ($sourceContent -eq $destContent) {
                $shouldCopy = $false
                Write-Host "Identique, ignoré : $($file.Name)"
            } else {
                Write-Host "Modifié, mise à jour : $($file.Name)"
            }
        } else {
            Write-Host "Nouveau fichier : $($file.Name)"
        }

        if ($shouldCopy) {
            Copy-Item $file.FullName $dest -Force
        }
    }
}

# COMMIT & PUSH AUTOMATIQUES
Set-Location $localRepoDir

git add bucket-public

# Vérifie s'il y a des changements à committer
$changes = git status --porcelain

if ($changes) {
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    git commit -m "🔄 Sync manifests publics Scoop - $timestamp"
    git push origin main
```

En local, il est nécessaire de cloner votre dépôt afin que le script puisse fonctionner correctement.

TEST:
configurer une identité Git

```
PS C:\Users\User> git config --global user.name "Nikas"
>> git config --global user.email "assoumani.moudjahidy@educ-valadon-limoges.fr"
```

copier le dépôt

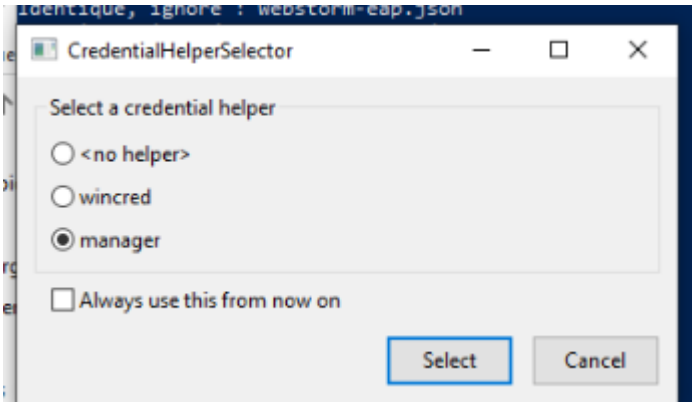
```
PS C:\Users\User> git clone https://[redacted]/pre-prod/my-custom-scoop-manifests.git
Cloning into 'my-custom-scoop-manifests'...
remote: Enumerating objects: 4147, done.
remote: Counting objects: 100% (4122/4122), done.
remote: Compressing objects: 100% (3889/3889), done.
remote: Total 4147 (delta 234), reused 4117 (delta 233), pack-reused 25 (from 1)
Receiving objects: 100% (4147/4147), 2.10 MiB | 12.49 MiB/s, done.
Resolving deltas: 100% (240/240), done.
Updating files: 100% (4128/4128), done.
```

exécuter le script

```
PS C:\Users\User> cd C:\Users\User\Downloads\
PS C:\Users\User\Downloads> powershell -ExecutionPolicy Bypass -File .\sync-public-buckets.ps1

--- Clonage de Main ---
Cloning into 'C:\Users\User\AppData\Local\Temp\scoop-public-sync\Main'...
remote: Enumerating objects: 1481, done.
remote: Counting objects: 100% (1481/1481), done.
remote: Compressing objects: 100% (1418/1418), done.
remote: Total 1481 (delta 144), reused 240 (delta 60), pack-reused 0 (from 0)
Receiving objects: 100% (1481/1481), 676.83 KiB | 2.11 MiB/s, done.
Resolving deltas: 100% (144/144), done.
Identique, ignoré : extensions.json
Identique, ignoré : settings.json
Identique, ignoré : 1password-cli.json
```

tout est ignoré car tout est déjà existant




appuyer sur select

```
[main d1aea2a] 🖹  Sync manifests publics Scoop - 2025-06-11 15:25:56
 25 files changed, 104 insertions(+), 104 deletions(-)
Enumerating objects: 54, done.
Counting objects: 100% (54/54), done.
Delta compression using up to 2 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (28/28), 5.35 KiB | 1.07 MiB/s, done.
Total 28 (delta 25), reused 0 (delta 0), pack-reused 0 (from 0)
To https://[redacted]/pre-prod/my-custom-scoop-manifests.git
   722d97f..d1aea2a  main -> main

🖹 Changements poussés vers GitLab.
```

Problème SSL rencontré sur windows pour pouvoir tester quand même j'ai désactiver temporairement la vérification avec : git config --global http.sslVerify false.

La création d'identité n’a pas non plus été prise en compte j’ai dû m'authentifier (ce qui n’est pas genant), peut-etre je l’ai mal faite tout simplement.

Déployer UnigetUI pour permettre aux enseignants de télécharger des logiciels de manière autonome, via un système de filtres.

La dernière fonctionnalité de notre bucket stocké en pré-production est de permettre le contrôle des installations via **UnigetUI**.

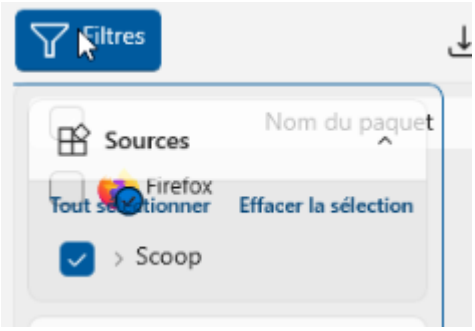
Explication :

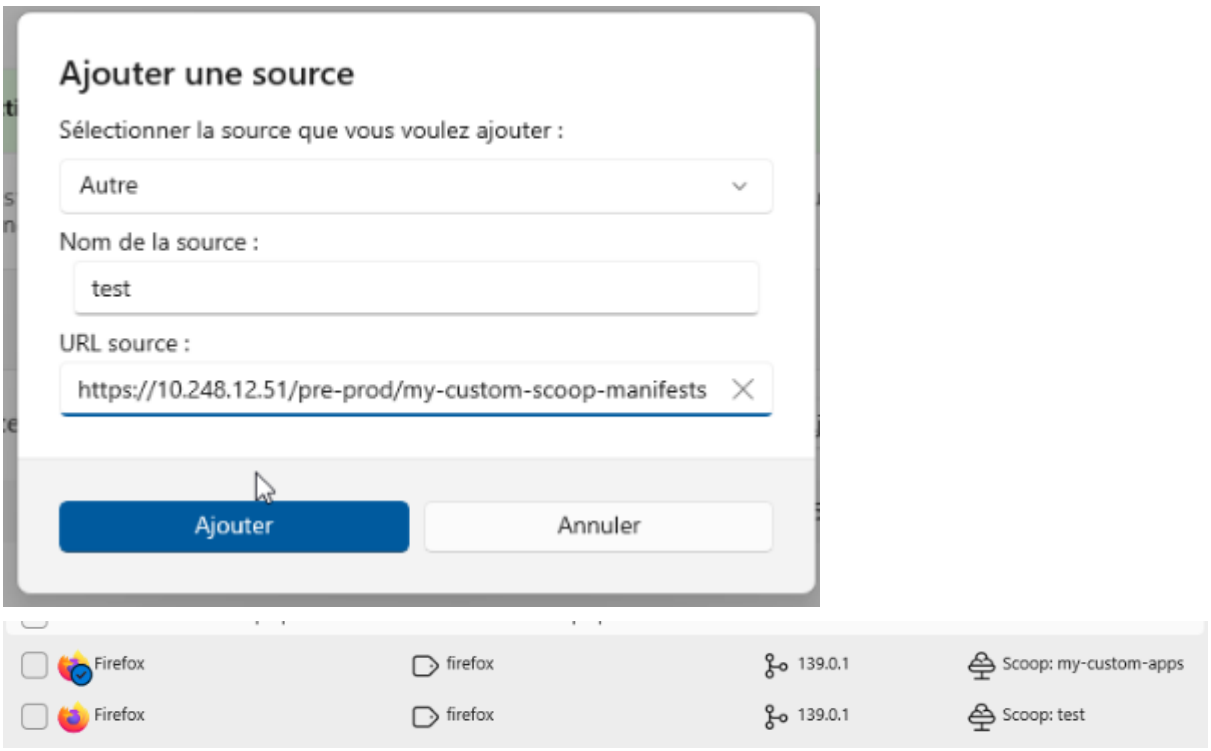
UnigetUI est une interface graphique qui permet de gérer plusieurs méthodes d’installation de logiciels. Dans notre cas, seule la méthode **Scoop** est conservée. On peut donc activer un filtre spécifique à Scoop dans l'interface.

Par défaut, UnigetUI ne s’appuie que sur le bucket `main` de Scoop, qui contient environ 1300 manifestes. Ce nombre est insuffisant pour répondre à nos besoins.
C’est là qu’intervient notre **bucket privé**, qui regroupe l’ensemble des buckets Scoop publics en un seul bucket centralisé. Ce dernier peut être ajouté à UnigetUI comme **source supplémentaire**.

⚠️ Remarque : comme UnigetUI ne gère qu’un seul bucket Scoop à la fois (il les ajoute dans le répertoire buckets de Scoop), notre regroupement en un unique bucket simplifie cette intégration.

Grâce à cette solution, les enseignants pourront installer plus de **4000 logiciels** en toute autonomie, directement depuis leurs espaces personnels, sans avoir besoin d’une intervention de l’administrateur.






Déployer UnigetUI pour permettre aux enseignants de télécharger des logiciels via Scoop, c'est une bonne idée pour l'autonomie et le contrôle, mais si la config doit être faite manuellement sur chaque poste, c'est vite ingérable à l'échelle d'un parc entier.

Solution :

Installe UnigetUI (s'il n'est pas déjà là).

Modifie les fichiers de configuration pour :

- Désactiver WinGet et les autres sources.
- N'activer que Scoop.
- Ajouter votre bucket Scoop personnalisé.

Impossible de trouver le fichier du coup j'ai été obligé d'en créer un en .json :

settings.json 406 B

```json
{
  "language": "fr",
  "theme": "system",
  "checkForUpdateAtStartup": true,
  "autoUpdateSources": true,
  "enabledSources": [
    "Scoop"
  ],
  "disabledSources": [
    "Winget",
    "Chocolatey",
    "PortableApps"
  ],
  "scoop": {
    "buckets": [
      {
        "name": "custom",
        "url": "https://[redacted]pre-prod/my-custom-scoop-manifests.git"
      }
    ]
  }
}
```

script powershell :

UnigetUIConfig.ps1 1.78 KiB

```powershell
# VARIABLES
$unigetInstallerUrl = "https://github.com/marticliment/UniGetUI/releases/latest/download/WingetUI.Installer.exe"
$unigetSetupPath = "$env:TEMP\UnigetUI-Setup.exe"
$installDir = "C:\Program Files\UnigetUI"

# URL du fichier settings.json hébergé sur ton GitLab
$configSource = "https://[redacted]/pre-prod/test-scoop/-/raw/main/settings.json"
$configDestDir = "$env:APPDATA\UnigetUI"
$configDestFile = Join-Path $configDestDir "settings.json"

# TÉLÉCHARGEMENT + INSTALLATION D'UNIGETUI
if (!(Test-Path "$installDir\UnigetUI.exe")) {
    Write-Host "Téléchargement de UnigetUI..."
    Invoke-WebRequest -Uri $unigetInstallerUrl -OutFile $unigetSetupPath

    Write-Host "Installation de UnigetUI..."
    Start-Process -FilePath $unigetSetupPath -ArgumentList "/VERYSILENT", "/NORESTART", "/DIR=`"$installDir`"" -Wait
    Remove-Item $unigetSetupPath -Force
} else {
    Write-Host "UnigetUI est déjà installé."
}

# Ajout manuel du bucket Scoop custom s'il n'existe pas déjà
$scoopBucketName = "custom"
$scoopBucketUrl = "https://[redacted]pre-prod/my-custom-scoop-manifests.git"

if (-not (scoop bucket list | Select-String -Pattern "^$scoopBucketName\s")) {
    Write-Host "Ajout du bucket Scoop personnalisé '$scoopBucketName'..."
    scoop bucket add $scoopBucketName $scoopBucketUrl
} else {
    Write-Host "Le bucket '$scoopBucketName' est déjà présent."
}

# CRÉATION DU DOSSIER CONFIG SI BESOIN
if (!(Test-Path $configDestDir)) {
    New-Item -Path $configDestDir -ItemType Directory -Force | Out-Null
}

# TÉLÉCHARGEMENT DU FICHIER DE CONFIGURATION settings.json
Write-Host "Téléchargement de la configuration personnalisée..."
Invoke-WebRequest -Uri $configSource -OutFile $configDestFile -UseBasicParsing

Write-Host "✅ Configuration UnigetUI copiée dans : $configDestFile"
```

Ce script PowerShell installe UniGetUI silencieusement s'il n'est pas déjà présent, ajoute un bucket Scoop personnalisé, puis télécharge un fichier de configuration settings.json depuis un GitLab privé et le place dans le dossier utilisateur (AppData\Roaming\UnigetUI).

Ce qui fonctionne bien :

- L’installation silencieuse de UniGetUI
- Le téléchargement et la copie du fichier de config
- L’ajout automatique du bucket Scoop personnalisé

Ce qui ne fonctionne toujours pas parfaitement :

- La désactivation effective des autres sources (Winget, Chocolatey, etc.)
- L’exécution automatique de l’app après install, qui gêne le script
- Le fait que l'app ignore settings.json (selon l'endroit ou moment où il est écrit)