Rakvere Ametikool

Remo Pasti

IT21

ÕPPELABORI VIRTUAALMASINATE HALDAMISE AUTOMATISEERIMINE

Eksamitöö

Juhendaja: Jelena Laidinen

Rakvere 2024

SISUKORD

1. EESSÕNA

Eksamitöö tegin üksinda, kuid teemakohast abi oli võimalik saada õpetajalt. Eksamitöö seisnes Hyper-V virtuaalmasinate, varundamiseks taastamiseks ja automatiseerimiseks. Eksamitöö teema valisin õpetaja poolt välja pakutud teemadest. Valisin selle teema, kuna see tundus huvitav ja tahtsin rohkem teada saada Powershell skriptimise ja virtuaalmasinate kohta, samuti olen ka varem sellega kokku puutunud.

Algul tutvusin eksamitöös vajamineva tarkvaraga iseseisvalt, uurides Hyper-V virtuaalmasinate haldamist ja sellega seonduvat. Siis hakkasin tegelema praktilise poolega, proovisin erinevaid meetodeid virtuaalmasinate varundamiseks skriptis. Kuna tegin sellist skripti esimest korda, siis oli see päris huvitav.

2. LAHENDITE JA TÄHISTE LOETELU

Hyper V - Microsofti poolt valmistatud virtualiseerimisplatvorm, mis võimaldab luua ja hallata virtuaalmasinaid.

VM - Virtuaalne masin.

VPN - Virtuaalne privaatvõrk, mis loob ühenduse kaugjuurdepääsuks.

Faili räsimine – Funktsioon, mis väljastab unikaalse väärtuse, mis vastab faili sisule

CLI - Käsurea kasutajaliides

CLI Lipp - käskluse argument

GUI - Graafiline kasutajaliides

SISSEJUHATUS

Antud eksamitöö keskendub Powershell skripti loomisele, mille eesmärgiks on tagada süstemaatiline ja usaldusväärne virtuaalmasinate varundamise automatiseerimine. Skripti peamine eesmärk on automatiseerida varundamisprotsessi luues dünaamiliselt kaustu kuupäeva põhjal ning tagades varundatud andmete terviklikkuse räsi funktsioonide abil. Lisaks kontrollib skript varasemalt salvestatud räsimiste abil varundatud andmete terviklikkust.

Selle töö käigus omandatud teadmised ja oskused hõlmavad struktureeritud ja dokumenteeritud varundamisprotsessi ja selle loomist Powershell abil muutujate, massiivide ja juhtlausetega töötamist Powershell keskkonnas ning andmete terviklikkuse tagamiseks räsi funktsioonide kasutamist. Antud eksamitöö pakub kindlat alust edasisteks arendusteks või kohandamiseks vastavalt konkreetsetele vajadustele ning demonstreerib võimalusi virtuaalmasinate varundamise automatiseerimiseks Powershell abil.

3. KASUTATAVA TEHNOLOOGIA KIRJELDUS

3.1 Windows Server 2022

Windows Server 2022 on Microsofti poolt välja antud serveri operatsioonisüsteem, mis on mõeldud ettevõtetele ja organisatsioonidele nende serverikeskkonna haldamiseks. See pakub stabiilset platvormi, turvafunktsioone ning võimalusi serveripõhiste rakenduste käitlemiseks.

Alternatiivideks oleks üks mitmest Linux Server operatsioon süsteemist näiteks Ubuntu server või Fedora server. Valisin Windows Server 2022, kuna see toetab Windows 11 kliente kõige paremini.

3.2 Hyper-V

Virtualiseerimisplatvorm, mis võimaldab kasutajatel luua ja hallata virtuaalseid masinaid. See on tihedalt integreeritud Windows Serveriga ning pakub ressursitõhusust, jõudlust ja paindlikkust

Alternatiivideks oleks VMware Workstation Player, KVM, Oracle Virtualbox. Valisin Hyper-V kuna koolis kasutatakse seda kõige rohkem ja seda on lihtne paigaldada.

3.3 Visual Studio Code

Microsofti poolt loodud arenduskeskkond, kus on võimalik Powershell skripte kirjutada ja käivitada. See pakub mugavat liidest, skriptimiseks vajalikke tööriistu ning võimaldab skriptide loomist ja testimist.

Alternatiivideks oleks Notepad++, Powershell ISE. Valisin Visual Studio Code kuna see on laialt levinud ja väga stabiilne.

3.4 Powershell

Powershell on Microsoft Windowsi skriptimis keel ja käitluskeskkond, mis on mõeldud süsteemihalduseks ja automatiseerimiseks Windowsi keskkonnas. See võimaldab kasutajatel automatiseerida erinevaid ülesandeid ning hallata süsteeme tõhusalt.

Alternatiivina oleks saanud kasutada Bash, Fish, Zsh ja sh. Valisin Powershell keele kuna see on sisse ehitatud Windowsi skriptimis keel.

3.5 Git

Git on hajus versioonihaldussüsteem, mis võimaldab arendajatel jälgida koodi muudatusi, hallata koodibaasi ja teha koostööd meeskonnaga. Alternatiividena oleks saanud kasutada Mercurial, Subversion (SVN). Valisin Git kuna Github vajas Git-i koodi varundamiseks.

3.6 Github

Github on platvorm, kus saab hoida Git-i projektide repositooriume, teha koostööd teiste arendajatega ning hallata tarkvaraarenduse tsüklit. Oleks saanud alternatiivina kasutada Gitlab, Bitbucket, Azure. Valisin Github-i kuna mul oli selles keskkonnas juba konto ja olin sellega varem kokku puutunud.

3.7 Task Scheduler

Windows opsüsteemi sisseehitatud tööriist, mis võimaldab kasutajatel automatiseerida ja ajastada erinevaid ülesandeid. Seda kasutatakse programmi käivitamiseks, skriptide või käskude käivitamiseks kindlatel aegadel või sündmuste korral, regulaarsete ülesannete täitmiseks ja muude automatiseeritud toimingute tegemiseks. Alternatiivina oleks saanud kasutada Automate, System Scheduler või VisualCron. Valisin just Windows Task Scheduler-i, kuna see on Windowsi programm, mis tuleb Windowsiga kaasa.

3.8 Group Policy Managment Console

On Microsoft Windowsi operatsioonisüsteemides kasutatav funktsioon, mis võimaldab süsteemiadministraatoritel hallata ja konfigureerida operatsioonisüsteemi. Alternatiivina oleks saanud kasutada Microsoft Endpoint Manager. Aga valisin Group Policy seetõttu, et olen sellega kokku puutunud.

4. LAHENDUSKÄIGU PLANEERING

Esimesena peaks hankima ISO-faili, mis sisaldab Windows Server 2022.

Järgmisena tuleks paigaldada operatsioonisüsteem riistvarale, mis vastab vähemalt miinimumnõuetele.

Hyper-V sisselülitamine on järgmine samm. Seda tuleks teha läbi "appwiz.cpl" programmi, valides "Lülita Windowsi funktsioonid sisse või välja" nupu alt. Järgmiseks tuleks valida Hyper-V teenus ja see sisse lülitada.

Kui eelnev on tehtud, peaks laadima alla Visual Studio Code Microsofti veebilehelt või ametlikult veebilehelt. Seejärel tuleks laadida Visual Studio Code'i Powershell laiendus.

Seejärel tuleb kirjutada skript, mis võimaldab Hyper-V virtuaalmasinate turvalist varundust võrgukettale.

Kui eelnev funktsionaalsus on skripti lisatud, tuleks leida viis, kuidas võrgukettale tehtud varundusest taastada virtuaalmasinad.

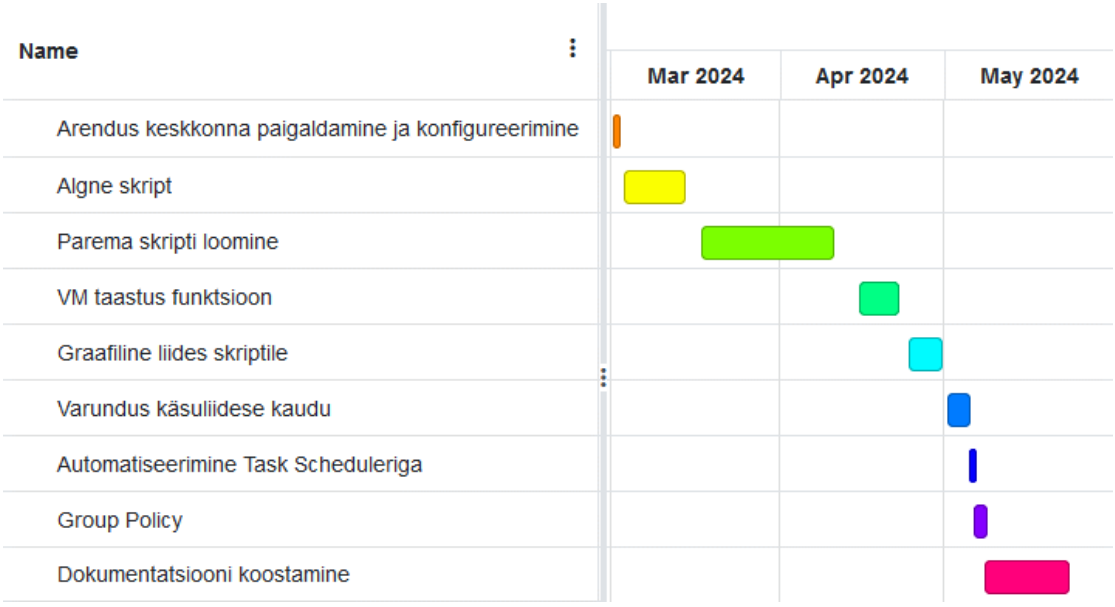
Et kasutajal oleks lihtsam, võiks juurde arendada graafilise liidese.

Kui graafiline liides on loodud, tuleks lisada nupud virtuaalmasinate varundamiseks ja taastamiseks. Selleks, et kasutajal oleks lihtsam varundust jälgida, võiks olla nupp, mis avab võrgukettalt kausta asukoha. [6]

Kui eelnev on tehtud, võiks lisada CLI käsuliidese, et hiljem saaks käivitada varundusprotsessi ja automatiseerida selle Task Schedule’is.

Järgmiseks tuleks kasutada Group Policy koos Task Schedule’iga, et käivitada skript CLI kaudu, eesmärgiga käivitada varundusfunktsionaalsus iga päev kell 16:45. Kui skript on jooksutatud, sulgeb Task Scheduler arvuti. [4]

Planeeritud tööde jaotust saab vaadata joonisel 1.



Joonis 1 - Ajakava

5. VAJALIKE RESSURSSIDE KIRJELDUS

Mul oli vaja teha eksamitööd kasutades kahte arvutit: üks neist oli koolis, teine minu kodus. Et saaksin kaugjuurdepääsu kooli arvutile, pidin looma ühenduse kooli VPN-tunnelisse. See võimaldas mul turvaliselt ühendada end kooli võrguga ka kodust. Nii sain vajalikke kooliressursse kasutada ja eksamitööd teha. VPN-tunnel tegi minu töö palju mugavamaks ja paindlikumaks, võimaldades töötada seal, kus mulle kõige paremini sobis.

5.1 Riistvara

Riistvara	Arvuti 1	Arvuti 2
CPU	Intel i7-2600	AMD Ryzen 5 5600X
RAM	16GB	32GB
SSD	512GB	1TB

5.2 Tarkvara

- Windows Server 2022
- Windows 10 Pro
- Hyper-V Server
- RDP klient
- Samba
- Powershell
- Powershell ISE
- Git
- Github
- Task Scheduler

5.3 Riistvara minimaalsed nõuded

- 64-bitine 4 tuumaga protsessor
- Vähemalt 8 GB mälu
- HDD vähemalt 64 GB

6. LAHENDUSE KIRJELDUS

6.1 Arendus keskkonna paigaldamine ja konfigureerimine

Esimese asjana laadisin koolist saadud arvutile operatsioonsüsteemi milleks oli Windows Server 2022.

Ning siis installisin Visual Studio Code ja paigaldasin laienduse, et saaksin kirjutada Powershell skripti Visual Studio-s.

6.2 Algse skripti loomine

Uurisin skriptide ja nende automatiseerimise kohta. Seejärel hakkasin skripti looma, kuid esimene katse testides ebaõnnestus. Analüüsisin olukorda ja leidsin mitmeid kohandamist vajavaid punkte. Seejärel lisasin skripti taastamisfunktsiooni, mis võimaldab valida varukoopiaid võrgukettalt. Lisasin ka räsi võrdlemise funktsionaalsuse, et tagada andmete terviklikkus ja autentsus. [1]

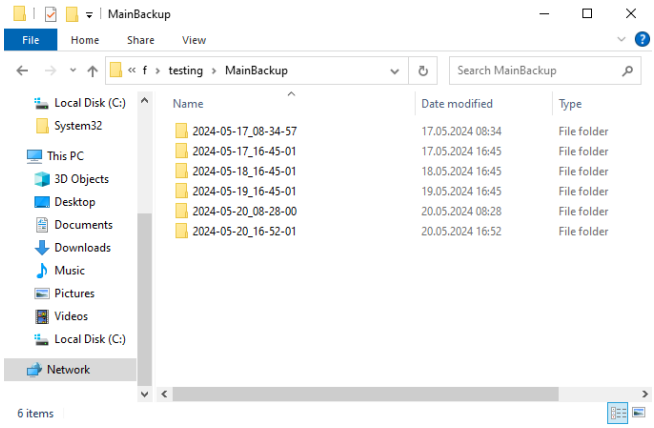
```
$networkFolderPath = "\\192.168.10.198\f\testing"
$vhdxFilesPath = "C:\Users\Public\Documents\Hyper-V\Virtual hard disks"
$mainBackupFolderPath = Join-Path $networkFolderPath "MainBackup"
$dateFolderPath = Join-Path $mainBackupFolderPath (Get-Date -Format "yyyy-MM-dd_HH-mm-ss")

if (Test-Path $networkFolderPath) {Write-Host "Backup script started"
    write-host "Stopping all Virtual Machines"
    Get-VM | Stop-VM -Force
    $archivePath = Join-Path $vhdxFilesPath "backup.zip"
    & "C:\Program Files\7-Zip\7z.exe" a -tzip $archivePath $vhdxFilesPath\*
    $archiveHash = Get-FileHash -Path $archivePath -Algorithm SHA256
    if (-not (Test-Path $mainBackupFolderPath)) {Write-Host "Making main backup folder"
        New-Item -ItemType Directory -Path $mainBackupFolderPath | Out-Null
        Write-Host "Main backup folder created."}
    else { Write-Host "Main backup folder already exists." }
    if (-not (Test-Path $dateFolderPath)) {
        New-Item -ItemType Directory -Path $dateFolderPath | Out-Null
        Write-Host "$dateFolderPath folder created."}
    else { Write-Host "$dateFolderPath folder already exists." }
    Copy-Item -Path $archivePath -Destination $dateFolderPath -Force -Verbose
    Write-Host "Archive copied to $networkFolderPath."
    $copiedArchiveHash = Get-FileHash -Path (Join-Path $dateFolderPath "backup.zip") -Algorithm SHA256
    if ($copiedArchiveHash.Hash -eq $archiveHash.Hash) { Write-Host "File integrity check passed for the archive." }
    else { Write-Host "File integrity check failed for the archive." }
    Copy-Item -Path $archivePath -Destination $dateFolderPath -Force -Verbose
    Write-Host "Archive copied to $networkFolderPath."
    $copiedArchiveHash = Get-FileHash -Path (Join-Path $dateFolderPath "backup.zip") -Algorithm SHA256
    if ($copiedArchiveHash.Hash -eq $archiveHash.Hash) { Write-Host "File integrity check passed for the archive." }
    else { Write-Host "File integrity check failed for the archive." }
    Remove-Item -Path $archivePath -Force
    $folders = Get-ChildItem -Path $mainBackupFolderPath -Directory
    if ($folders.Count -gt 5) {$oldestFolder = $folders | Sort-Object CreationTime | Select-Object -First 1
        Write-Host "Deleting the oldest folder: $($oldestFolder.Name)"
        Remove-Item -Path $oldestFolder.FullName -Recurse -Force}
    else {Write-Host "There are 5 or fewer folders in MainBackup. No action taken."}}
else { Write-Host "Network folder is not accessible." }
```

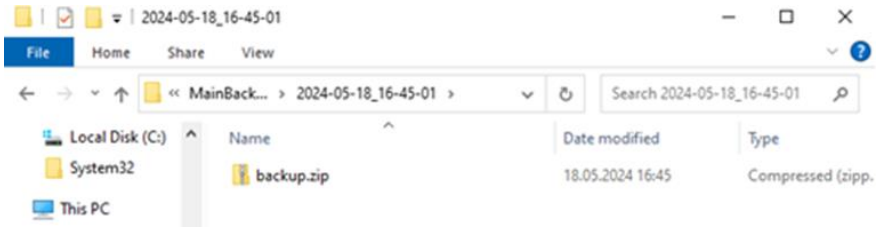
Joonis 2 - Algne skript

6.3 *Parema skripti loomine*

Peale eelmist ebaõnnestumist hakkasin sarnaseid skripte uurima ning Microsofti Powershell dokumentatsiooni lugema. Peale dokumentatsiooni lugemist sain teada, et Hyper-V virtuaalseid masinaid on võimalik eksportida käsuliidese kaudu. Uurisin palju juurde ning täiendasin oma vana skripti. Lisasin ka räsi funktsiooni, et kontrollida faili terviklikust peale võrgukettale kopeerimist. Kogusin uusi teadmisi ning siis tegin uue skripti, millele lisasin veel uusi funktsioone. Kui proovisin uut skripti, olin rahul kui nägin, et kõik töötab sujuvalt. See skript kõigepealt loob kuupäevaga kausta (vt Joonis 3) ja paneb kokku pakitud virtuaalmasinate failid (vt Joonis 4). [2]



Joonis 3 - Kuupäevaga kaustad



Joonis 4 - Virtuaalmasina arhiiv

Lõplik skript on toodud välja lisal 4.

6.3.1 *VM Taastus funktsioon*

Siis hakkasin taastamisfunktsiooni lisamisega pihta. Tegin skripti nii, et saab valida varukoopiaid võrgukettalt. Selleks, et varukoopiate terviklikust kontrollida lisasin skripti ka räsi arvutamise ja võrdlemise funktsiooni. See tagab, et andmed pole taastamise käigus muutunud või rikutud. VM taastus funktsiooni kohta saab näha lisa 2.

6.3.2 Graafiline liides skriptile

Peale parema skripti loomist ja taastus funktsiooni lisamist lisasin graafilise liidese, et tulevatel kasutajatel oleks seda kergem kasutada. Siis tuli mul idee, et kasutaja saaks ligi tähtsatele kaustadele. Lisasin graafilisse liidesesse nupud kus saab ligi ajutisele varundus kaustale ja lõplikule võrguketta kaustale. Graafilise liidese kohta on pilt lisas 3. [5]

6.3.3 varundus käsuliidese kaudu

Peale skripti natukese ümber tegemise oli võimalik lisada funktsionaalsus käsurea lipukese kujul. See tähendab, et kui lisada -cli lipp käsu lõppu siis on võimalik jooksutada varunduse funktsioon ilma kasutaja sekkumiseta. Seda funktsionaalsust on vaja selleks, et oleks võimalik varundus protsessi hiljem automatiseerida (vt Joonis 5).

```
  1   [CmdletBinding()]
  2 > param ([switch]$cli)···
 99 > if ($cli) { Backup } else {···
149   }
150
```

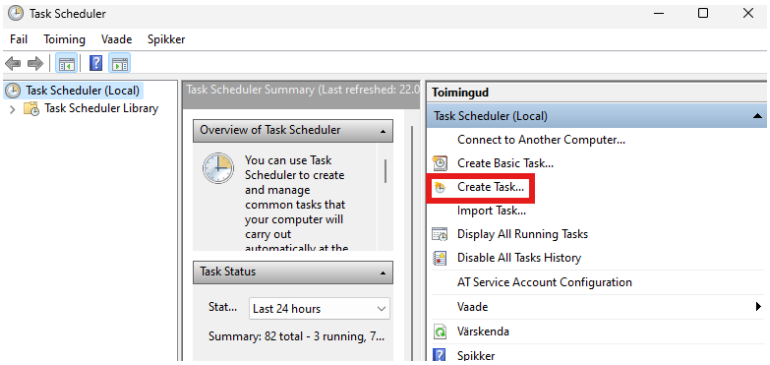
Joonis 5 - If funktsioon -cli lipu jaoks

6.4 Skripti jooksutamine Task Scheduler-i *või Group Policy*-ga

6.4.1 Task Scheduler

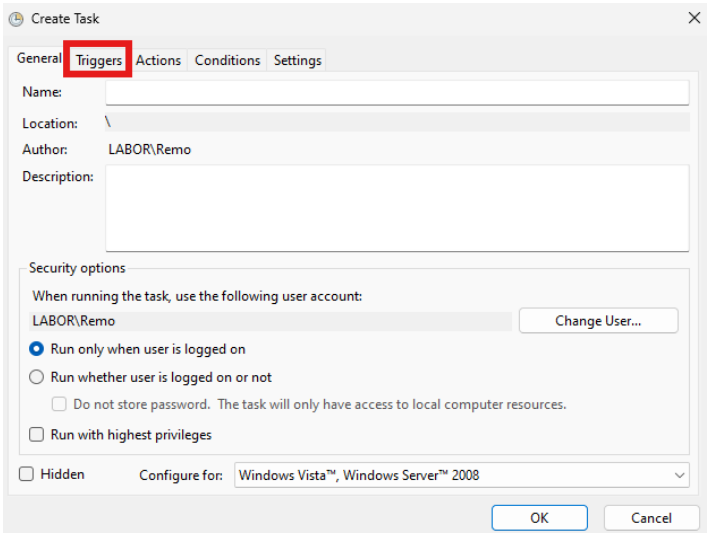
Selleks, et üks haaval automatiseerida seda protsessi klient arvutitel tuleb järgida järgmisi samme:

1. Avasin Task Scheduler programmi ja vajutasin parempoolselt menüü ribalt *create task (vt Joonis 6).*



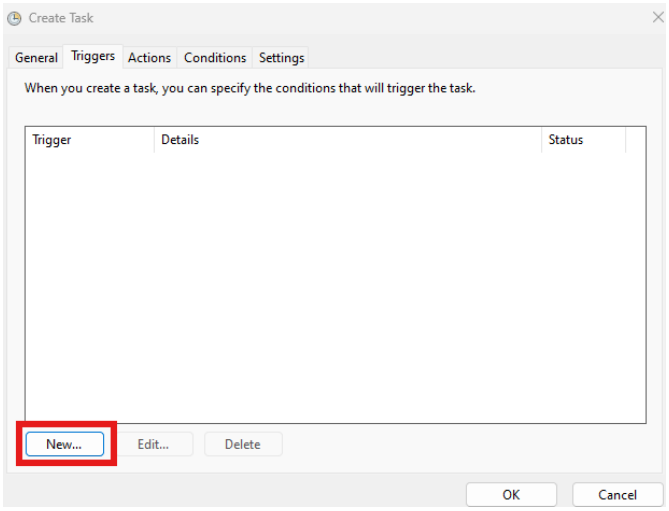
Joonis 6 - Uue tegevuse lisamine

2. Järgmisena liikusin *triggers* menüü sakki (vt Joonis 7).



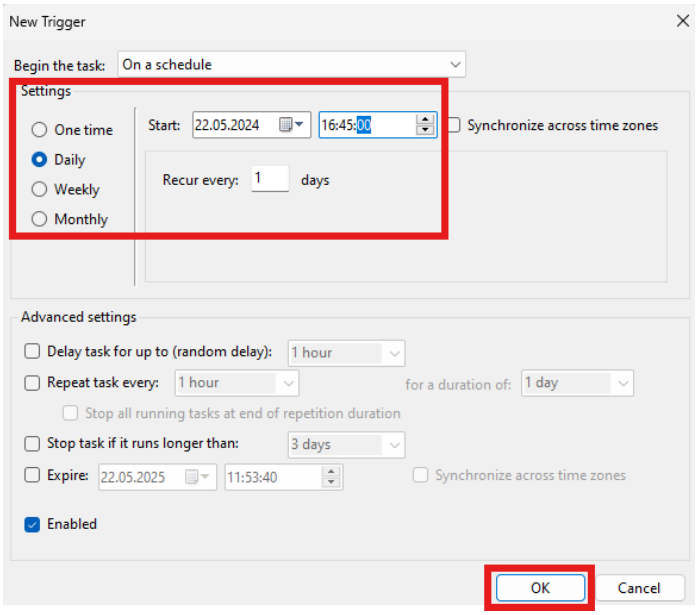
Joonis 7 - Uue käivitus põhjuse lisamine

3. Vajutasin *new* nupule (vt Joonis 8).



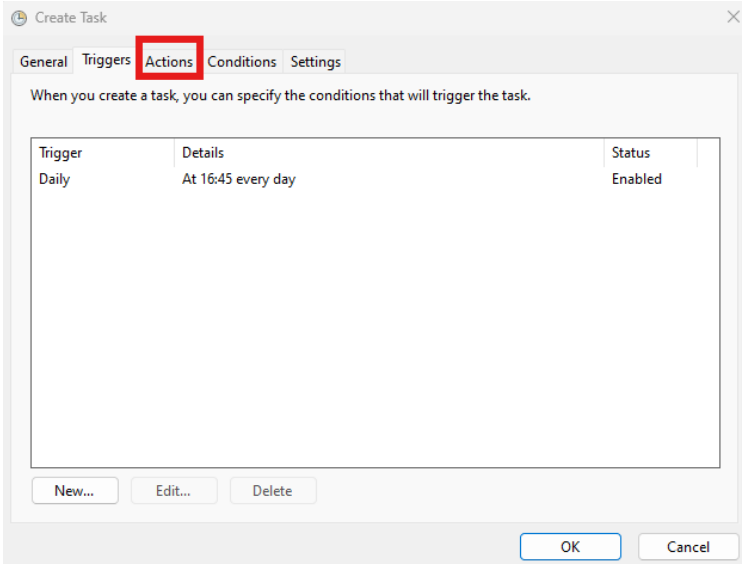
Joonis 8 - Uue käivituse lisamine

4. Järgmisena valisin *daily* raadio nupu ja lisasin sobiva kellaaja ja vajutasin ok nupule (vt Joonis 9).



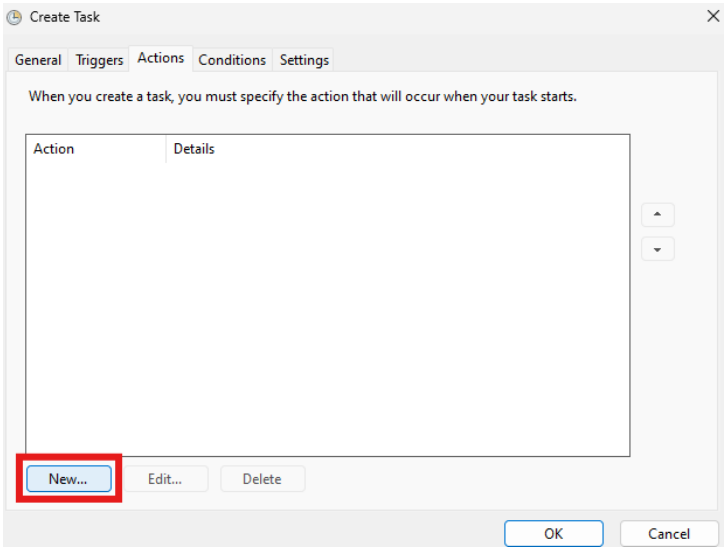
Joonis 9 - Sobiva kellaaja lisamine

5. Nüüd käivitub skript sellel kellaajal (10).



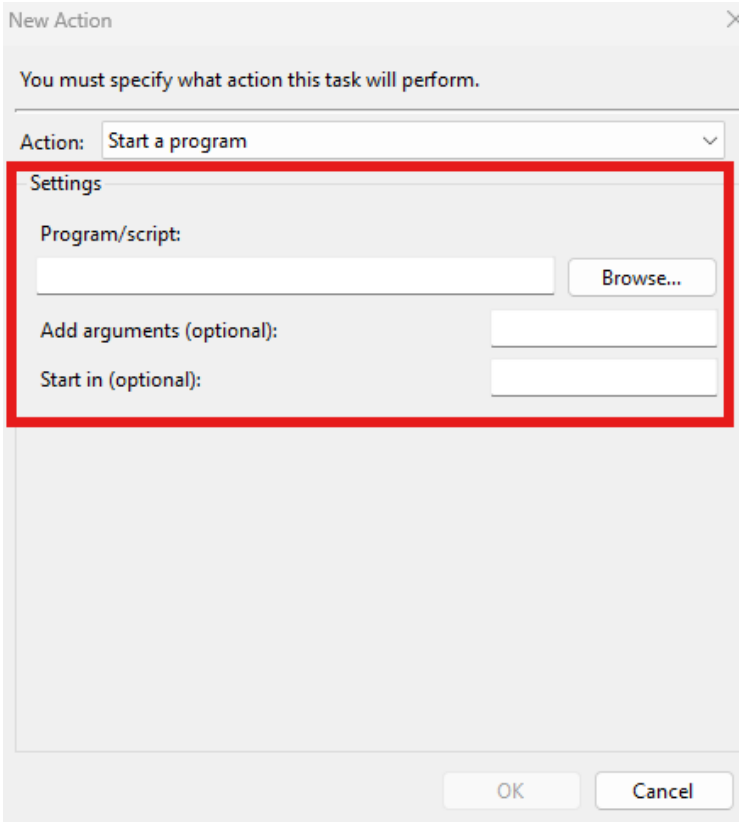
Joonis 10 - Tegevuste lisamine

6. Siis vajutasin *actions* menüü sakile (11).



Joonis 11 - Lisasin tegevuse

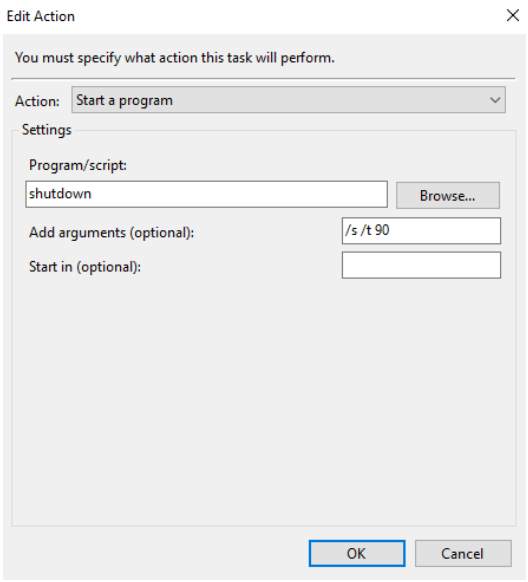
7. Siis vajutasin *new action* (12).



Joonis 12 - Skripti lisamine tegevuseks

8. Siis lisasin *program/script*-i lahtrisse Powershell. Järgmisena lisasin *add arguments* alla (-file "C:\Users\Administrator\Desktop\backup-script\vm-backup.ps1" -cli), et Powershell jooksutaks minu skripti. Lõpuks vajutasin ok nupule (vt Joonis 13).

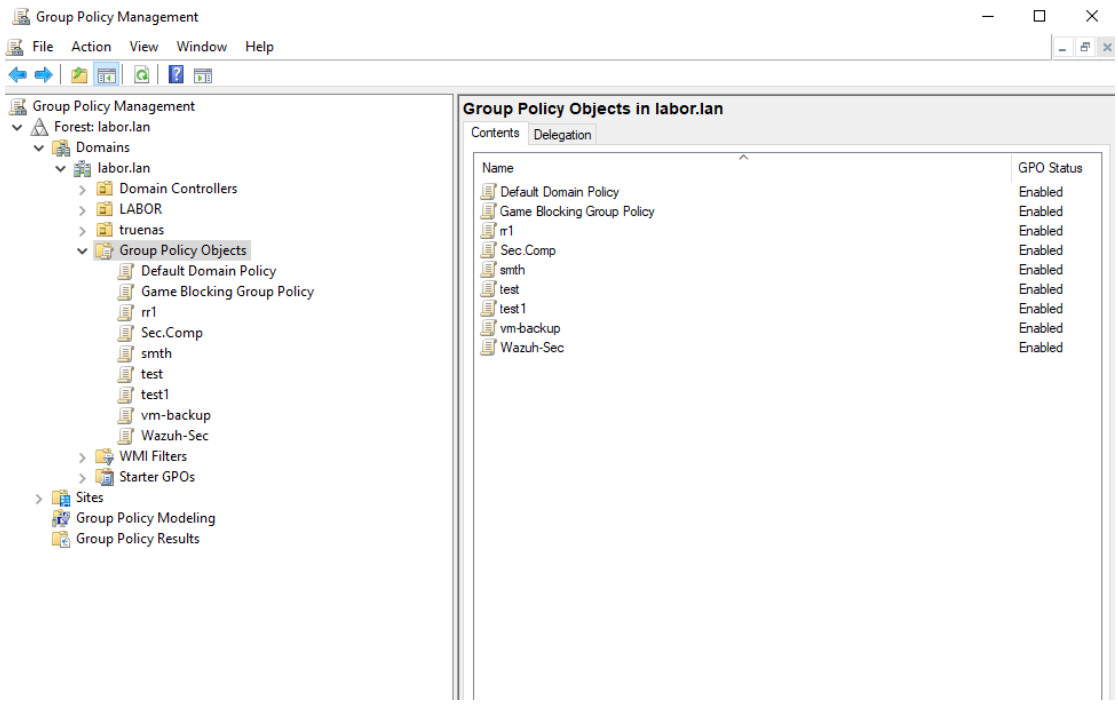
9. Kui eelnevaga sain tehtud siis lisasin uue tegevuse. See tegevus paneb arvutid kinni peale eelneva tegevuse lõpetamist (vt Joonis 13).



Joonis 13 - Lisasin arvuti välja lülituse

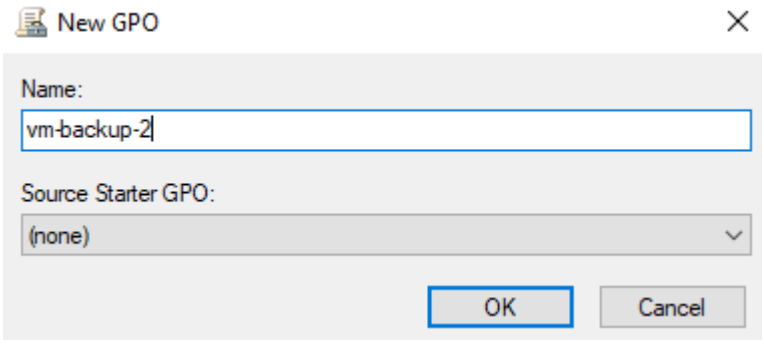
6.4.2 Group Policy

1. Avasin Group Policy Management rakenduse
2. Liikusin siia koolis kasutatava poliisi kausta (vt Joonis 14).



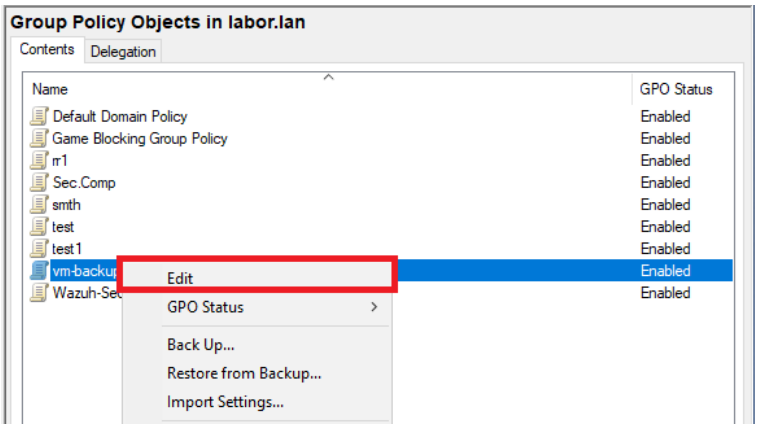
Joonis 14 - Õigesse kohta poliisi lisamine

3. Siis tegin uue gruppi poliisi vajutades parema klahviga parempoolsele paneelile ja valides *new* menüü valikut. Lisasin nime ja vajutasin ok nupule (vt Joonis 15).



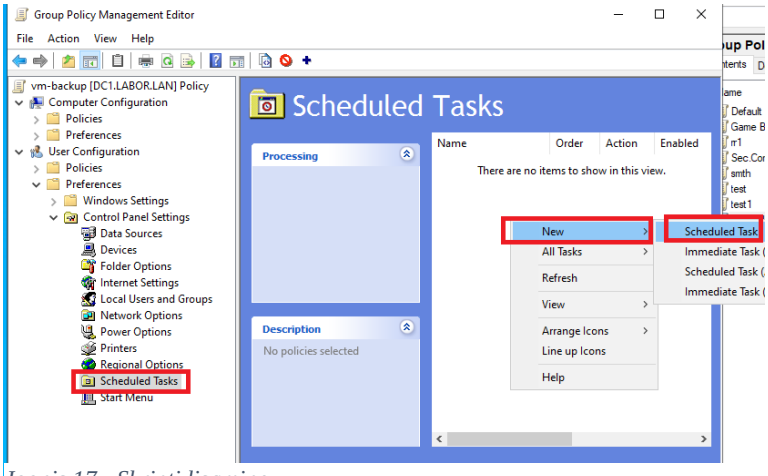
Joonis 15 - Uue poliisi lisamine

4. Nüüd kui parema klahviga vajutada äsja loodud poliisile peab valima *edit* menüü valikut, et lisada uus ajastatud skript (vt Joonis 16).



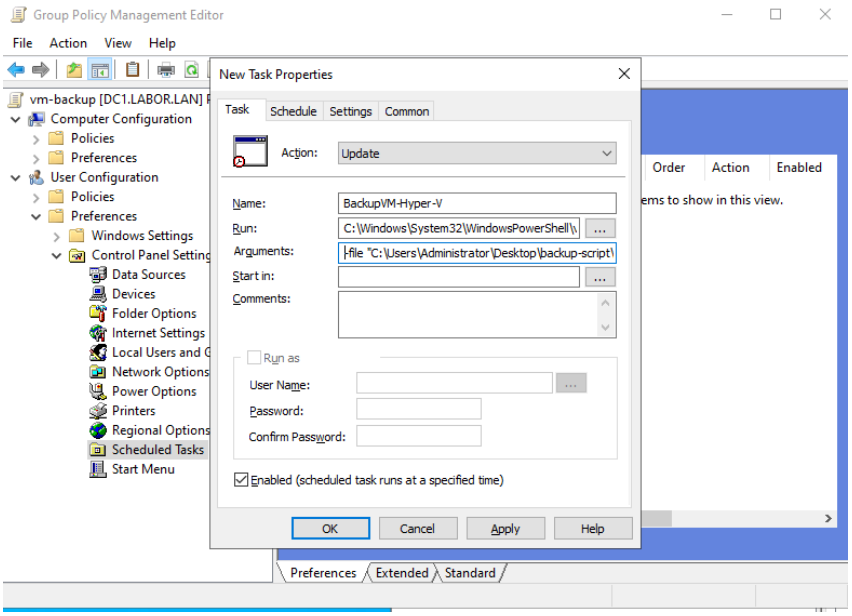
Joonis 16 - Uue ajastatud skripti valimine

5. Järgmisena navigeerisin Scheduled Tasks alla ja parema klahviga vajutasin paremale paneelile, et lisada skript (vt Joonis 17).



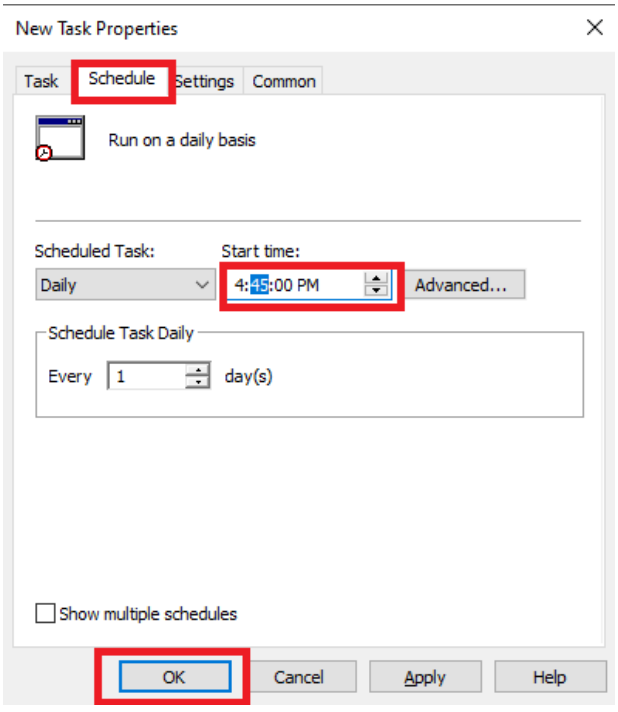
Joonis 17 - Skripti lisamine

6. Kõigepealt pidin nime lisama tegevusele. Järgmisena lisasin *run* kasti Powershell programmi ja *arguments* kasti (-file "C:\Users\Administrator\Desktop\backup-script\vm-backup.ps1" -cli) (vt Joonis 18).



Joonis 18 - Tegevuse lisamine ja argumentide lisamine

7. Siis läksin *Schedule* sakk menüüle ja lisasin tegevuse käivitamiseks sageduse. Lõpuks vajutasin *OK* nupule (vt Joonis 19).

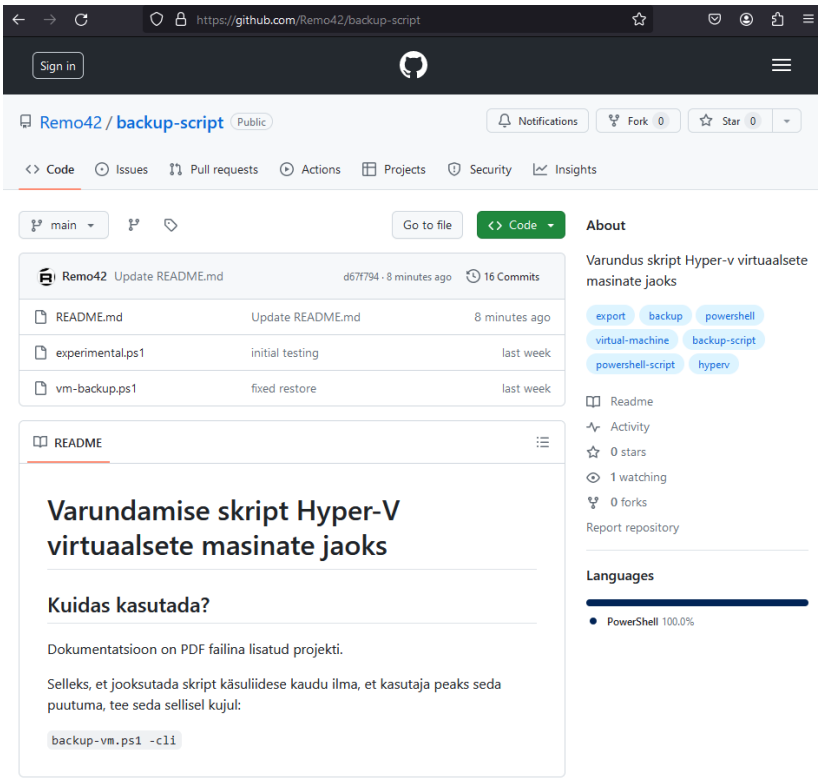


Joonis 19 - Tegevuse ajastamine

8. Nüüd peaks poliis kehtima domeenis olevatele arvutitele peale arvutite taaskäivitust või peale *gpupdate /force* käsu jooksutamist.

6.5 Skripti hoiustamine Github-is

Mul tuli hea mõte, et kui skript peaks kaduma minema siis on seda alati võimalik uuesti alla laadida ja dokumentatsiooni uuesti vaadata. Selleks, et seda teha tegin endale Github konto ja tegin projekti. Kui tühi projekt sai tehtud, laadisin skripti faili sellesse projekti. Projekti link on https://github.com/Remo42/backup-script (vt Joonis 20).



Joonis 20 - Skripti lisamine Github-i

KOKKUVÕTE

Töö tulemusena valmis skript, millel on graafiline liides, võimaldades Windows Hyper-V virtuaalmasinate varundamist. Seda funktsionaalsust saab käivitada nii graafilisest liidesest kui ka käsurealt. Lisaks saab graafilise liidese abil virtuaalmasinaid taastada kohalikule arvutile.

Töö käigus omandasin uusi teadmisi Powershell skriptide kohta ja Windowsi käsurea tööriistade kohta. Lõplik lahendus lihtsustab Hyper-V virtuaalmasinate varundust ja taastamist. Nüüd on olemas lihtne viis, kuidas õpilased saavad oma Hyper-V virtuaalmasinaid varundada ja taastada.

Need kogemused lihtsustavad tulevikus sarnaste skriptide kirjutamist. Need kogemused aitasid arendada minu probleemilahendusoskusi ja suurendada enesekindlust keerukate ülesannete lahendamisel.

KASUTATUD KIRJANDUS

1. Veebileht. Microsoft. Dokumentatsioon Powershell skriptile [Vaadatud 01.03.2024] https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-5.1

2. Veebileht Microsoft. Powershell dokumentatsioon Hyper-V [Vaadatud 17.03.2024] https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/deploy/export-and-import-virtual-machines

3. Veebileht Microsoft. Õppematerjal Powershell skriptile [Vaadatud 05.01.2024] https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-filehash?view=powershell-7.4

4. Artikkel veebis. Brien Posey How To Automate Powershell Scripts With Windows Task Scheduler [Vaadatud 05.05.2024]

https://www.itprotoday.com/powershell/how-automate-powershell-scripts-windows-task-scheduler

5. Õppematerjal Github repositoorium [Vaadatud 04.24.2024]

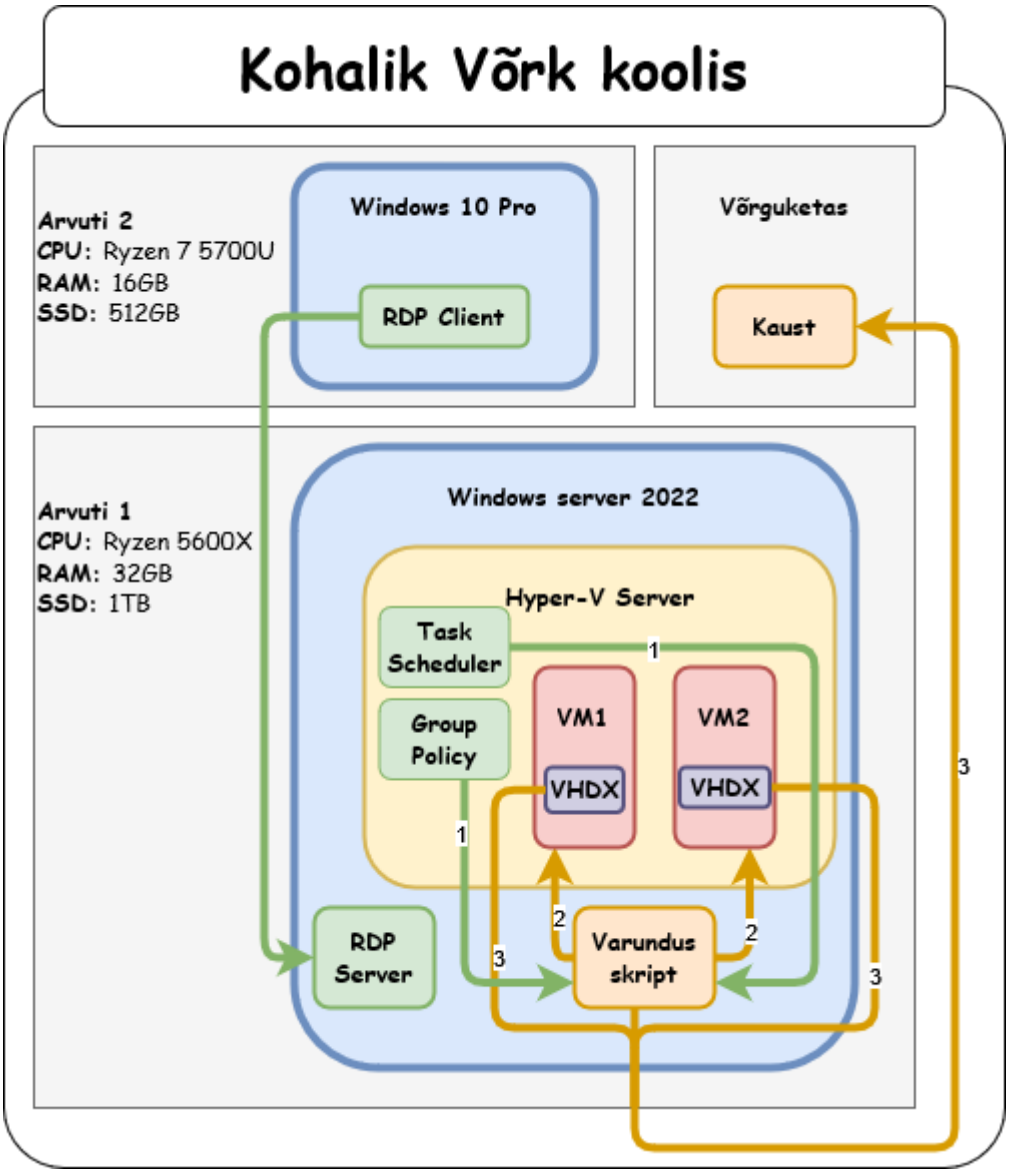
https://github.com/Gren-95/Windows-toolbox

6. Õppematerjal Github repositoorium [Vaadatud 04.24.2024]

https://github.com/Gren-95/Windows-School-Sample

LISAD

Lisa 1 – Joonis skripti kohta



Lisa 2 – Taastus funktsioon

```
function Restore {$networkFolderPath = "\\192.168.10.198\f\testing"
    $confirm = [System.Windows.Forms.MessageBox]::Show("This will STOP all running VMs, DELETE existing VMs, and RESTORE from a backup. Continue?",
        "Restore Confirmation",[System.Windows.Forms.MessageBoxButtons]::YesNo,[System.Windows.Forms.MessageBoxIcon]::Warning)
    if ($confirm -eq "Yes") {$mainbackup = Join-Path $networkFolderPath "MainBackup"
        $backupFolders = Get-ChildItem -Path $mainbackup -Directory -ErrorAction SilentlyContinue
        if ($backupFolders.Count -eq 0) {[System.Windows.Forms.MessageBox]::Show("Network folder is not accessible or no folders found.", "Error", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Error)
        return} Show-RestoreForm -BackupFolders $backupFolders}
}
function Show-RestoreForm {param ([System.IO.DirectoryInfo[]]$BackupFolders)
    $RestoreForm = New-Object System.Windows.Forms.Form
    $RestoreForm.StartPosition = "CenterScreen"
    $RestoreForm.Size = "400,300"
    $RestoreForm.FormBorderStyle = 'FixedDialog'
    $RestoreForm.MaximizeBox = $false
    $RestoreForm.BackColor = [System.Drawing.Color]::FromArgb(31, 31, 31)
    $RestoreForm.Text = "Restore from Backup"
    $label = New-Object System.Windows.Forms.Label
    $label.Location = New-Object System.Drawing.Point(20, 10)
    $label.Size = New-Object System.Drawing.Size(300, 20)
    $label.Text = "Select a backup to restore:"
    $label.ForeColor = [System.Drawing.Color]::White
    $RestoreForm.Controls.Add($label)
    $listBox = New-Object System.Windows.Forms.ListBox
    $listBox.Location = New-Object System.Drawing.Point(20, 40)
    $listBox.Size = New-Object System.Drawing.Size(300, 150)
    $listBox.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $listBox.ForeColor = [System.Drawing.Color]::White
    foreach ($folder in $BackupFolders) { $listBox.Items.Add($folder.Name) }
    $RestoreForm.Controls.Add($listBox)
    $okButton = New-Object System.Windows.Forms.Button
    $okButton.Location = New-Object System.Drawing.Point(150, 200)
    $okButton.Size = New-Object System.Drawing.Size(75, 25)
    $okButton.Text = "OK"
    $okButton.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $okButton.ForeColor = [System.Drawing.Color]::White
    $okButton.Add_Click({$selectedFolder = $listBox.SelectedItem
            if ($selectedFolder) { RestoreFromFolder -folderName $selectedFolder }
            else { [System.Windows.Forms.MessageBox]::Show("Please select a backup folder.", "Error", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Error) }})
    $RestoreForm.Controls.Add($okButton)
    $cancelButton = New-Object System.Windows.Forms.Button
    $cancelButton.Location = New-Object System.Drawing.Point(250, 200)
    $cancelButton.Size = New-Object System.Drawing.Size(75, 25)
    $cancelButton.Text = "Cancel"
    $cancelButton.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $cancelButton.ForeColor = [System.Drawing.Color]::White
    $cancelButton.Add_Click({ $RestoreForm.Close() })
    $RestoreForm.Controls.Add($cancelButton)
    function RestoreFromFolder {param ([string]$folderName)
        $tempfolder = "C:\hyper-v"
        $selectedFolder = Join-Path $mainbackup $folderName
        $archivePath = Join-Path $selectedFolder "backup.zip"
        Get-VM | Stop-VM -Force
        Get-VM | Remove-VM -Force
        if (Test-Path $tempfolder) {Remove-Item -Force -Recurse $tempfolder}
        mkdir $tempfolder
        Copy-Item $archivePath $tempfolder
        & "C:\Program Files\7-Zip\7z.exe" x "$tempfolder\backup.zip" "-o$tempfolder" -y
        $VMFiles = Get-ChildItem -Path "$tempfolder\*\Virtual Machines" -Filter *.vmcx -Recurse
        echo "Importing VM-s"
        foreach ($VMFile in $VMFiles) {Import-VM -Path $VMFile.FullName}
        [System.Windows.Forms.MessageBox]::Show("Restore completed successfully.", "Success", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Information)}
    [void]$RestoreForm.ShowDialog()
}
```

Lisa 3 – Graafiline liides

```
$App = New-Object System.Windows.Forms.Form
$App.StartPosition = "CenterScreen"
$App.Size = "270,260"
$App.FormBorderStyle = 'Fixed3D'
$App.MinimizeBox = $false
$App.MaximizeBox = $false
$App.BackColor = [System.Drawing.Color]::FromArgb(31, 31, 31)
$App.Text = "Hyper-V Backup"
$label = New-Object System.Windows.Forms.Label
$label.Location = New-Object System.Drawing.Point(25, 25)
$label.Font = New-Object System.Drawing.Font('verdana', 16)
$label.AutoSize = $true
$label.ForeColor = [System.Drawing.Color]::White
$label.Text = "Hyper-V Backup"
$App.Controls.Add($label)
$backup_button = New-Object System.Windows.Forms.Button
$backup_button.Location = New-Object System.Drawing.Point(25, 100)
$backup_button.Size = New-Object System.Drawing.Size(200, 25)
$backup_button.Text = "BACKUP"
$backup_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
$backup_button.ForeColor = [System.Drawing.Color]::White
$backup_button.Add_Click({ & { Backup } })
$App.Controls.Add($backup_button)
$restore_button = New-Object System.Windows.Forms.Button
$restore_button.Location = New-Object System.Drawing.Point(25, 125)
$restore_button.Size = New-Object System.Drawing.Size(200, 25)
$restore_button.Text = "RESTORE"
$restore_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
$restore_button.ForeColor = [System.Drawing.Color]::White
$restore_button.Add_Click({ & { Restore } })
$App.Controls.Add($restore_button)
$browse_button = New-Object System.Windows.Forms.Button
$browse_button.Location = New-Object System.Drawing.Point(25, 150)
$browse_button.Size = New-Object System.Drawing.Size(200, 25)
$browse_button.Text = "BROWSE BACKUPS"
$browse_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
$browse_button.ForeColor = [System.Drawing.Color]::White
$browse_button.Add_Click({ explorer.exe $networkFolderPath })
$App.Controls.Add($browse_button)
$browse_button2 = New-Object System.Windows.Forms.Button
$browse_button2.Location = New-Object System.Drawing.Point(25, 175)
$browse_button2.Size = New-Object System.Drawing.Size(200, 25)
$browse_button2.Text = "BROWSE TEMPORARY FILES"
$browse_button2.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
$browse_button2.ForeColor = [System.Drawing.Color]::White
$browse_button2.Add_Click({ explorer.exe $tempfolder })
$App.Controls.Add($browse_button2)
$App.Add_Shown({ $App.Activate() })
[void] $App.ShowDialog()
```

Lisa 4 – Terve skript

```
[CmdletBinding()]
param ([switch]$cli)
    $hyperVInstalled = Get-WindowsFeature -Name Hyper-V | Where-Object { $_.Installed }
    if (-not $hyperVInstalled) {Write-Host "Hyper-V is not installed. This script requires Hyper-V to be installed to run.";Exit}
    [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")
    [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")
    $networkFolderPath = "\\192.168.10.198\f\testing"
    $tempfolder = "C:\tempbackup"
    function Backup {$tempfolder = "C:\tempbackup"
        if (Test-Path $networkFolderPath) {Write-Host "Backup script started"
            Write-Host "Stopping all Virtual Machines"
            Get-VM | Stop-VM -Force
            if (Test-Path $tempfolder) {Get-VM | Stop-VM -Force
                Remove-Item -Force -Recurse $tempfolder}
            mkdir $tempfolder
            Export-VM * $tempfolder
            $archivePath = Join-Path $tempfolder "backup.zip"
            & "C:\Program Files\7-Zip\7z.exe" a -tzip $archivePath $tempfolder\*
            $archiveHash = Get-FileHash -Path $archivePath -Algorithm SHA256
            $mainBackupFolderPath = Join-Path $networkFolderPath "MainBackup"
            if (-not (Test-Path $mainBackupFolderPath)) {New-Item -ItemType Directory -Path $mainBackupFolderPath | Out-Null}
            $dateFolderPath = Join-Path $mainBackupFolderPath (Get-Date -Format "yyyy-MM-dd_HH-mm-ss")
            if (-not (Test-Path $dateFolderPath)) {New-Item -ItemType Directory -Path $dateFolderPath | Out-Null}
            Copy-Item -Path $archivePath -Destination $dateFolderPath -Force -Verbose
            Write-Host "Archive copied to $networkFolderPath."
            $copiedArchiveHash = Get-FileHash -Path (Join-Path $dateFolderPath "backup.zip") -Algorithm SHA256
            if ($copiedArchiveHash.Hash -eq $archiveHash.Hash) {Write-Host "File integrity check passed for the archive."}
            else {Write-Host "File integrity check failed for the archive."}

            Remove-Item -Path $archivePath -Force
            $folders = Get-ChildItem -Path $mainBackupFolderPath -Directory
            if ($folders.Count -gt 5) {$oldestFolder = $folders | Sort-Object CreationTime | Select-Object -First 1
                Write-Host "Deleting the oldest folder: $($oldestFolder.Name)"
                Remove-Item -Path $oldestFolder.FullName -Recurse -Force}
            else {Write-Host "There are 5 or fewer folders in MainBackup. No action taken."}} else {Write-Host "Network folder is   not accessible."}}
    function Restore {$networkFolderPath = "\\192.168.10.198\f\testing"
        $confirm = [System.Windows.Forms.MessageBox]::Show("This will STOP all running VMs, DELETE existing VMs, and RESTORE from a     backup. Continue?",
            "Restore Confirmation",
            [System.Windows.Forms.MessageBoxButtons]::YesNo,
            [System.Windows.Forms.MessageBoxIcon]::Warning)
        if ($confirm -eq "Yes") {$mainbackup = Join-Path $networkFolderPath "MainBackup"
            $backupFolders = Get-ChildItem -Path $mainbackup -Directory -ErrorAction SilentlyContinue
            if ($backupFolders.Count -eq 0) {[System.Windows.Forms.MessageBox]::Show("Network folder is not accessible or no    folders found.", "Error", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Error)
                return} Show-RestoreForm -BackupFolders $backupFolders}}
    function Show-RestoreForm {param ([System.IO.DirectoryInfo[]]$BackupFolders)
        $RestoreForm = New-Object System.Windows.Forms.Form
        $RestoreForm.StartPosition = "CenterScreen"
        $RestoreForm.Size = "400,300"
        $RestoreForm.FormBorderStyle = 'FixedDialog'
```

Lisa 4 järg

```
        $RestoreForm.MaximizeBox = $false
        $RestoreForm.BackColor = [System.Drawing.Color]::FromArgb(31, 31, 31)
        $RestoreForm.Text = "Restore from Backup"
        $label = New-Object System.Windows.Forms.Label
        $label.Location = New-Object System.Drawing.Point(20, 10)
        $label.Size = New-Object System.Drawing.Size(300, 20)
        $label.Text = "Select a backup to restore:"
        $label.ForeColor = [System.Drawing.Color]::White
        $RestoreForm.Controls.Add($label)
        $listBox = New-Object System.Windows.Forms.ListBox
        $listBox.Location = New-Object System.Drawing.Point(20, 40)
        $listBox.Size = New-Object System.Drawing.Size(300, 150)
        $listBox.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
        $listBox.ForeColor = [System.Drawing.Color]::White
        foreach ($folder in $BackupFolders) { $listBox.Items.Add($folder.Name) }
        $RestoreForm.Controls.Add($listBox)
        $okButton = New-Object System.Windows.Forms.Button
        $okButton.Location = New-Object System.Drawing.Point(150, 200)
        $okButton.Size = New-Object System.Drawing.Size(75, 25)
        $okButton.Text = "OK"
        $okButton.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
        $okButton.ForeColor = [System.Drawing.Color]::White
        $okButton.Add_Click({$selectedFolder = $listBox.SelectedItem
                if ($selectedFolder) { RestoreFromFolder -folderName $selectedFolder }
                else { [System.Windows.Forms.MessageBox]::Show("Please select a backup folder.",
                "Error", [System.Windows.Forms.    MessageBoxButtons]::OK, [System.Windows.Forms.
                MessageBoxIcon]::Error) }})
        $RestoreForm.Controls.Add($okButton)
        $cancelButton = New-Object System.Windows.Forms.Button
        $cancelButton.Location = New-Object System.Drawing.Point(250, 200)
        $cancelButton.Size = New-Object System.Drawing.Size(75, 25)
        $cancelButton.Text = "Cancel"
        $cancelButton.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
        $cancelButton.ForeColor = [System.Drawing.Color]::White
        $cancelButton.Add_Click({ $RestoreForm.Close() })
        $RestoreForm.Controls.Add($cancelButton)
        function RestoreFromFolder {param ([string]$folderName)
            $tempfolder = "C:\hyper-v"
            $selectedFolder = Join-Path $mainbackup $folderName
            $archivePath = Join-Path $selectedFolder "backup.zip"
            Get-VM | Stop-VM -Force
            Get-VM | Remove-VM -Force
            if (Test-Path $tempfolder) {Remove-Item -Force -Recurse $tempfolder}
            mkdir $tempfolder
            Copy-Item $archivePath $tempfolder
            & "C:\Program Files\7-Zip\7z.exe" x "$tempfolder\backup.zip" "-o$tempfolder" -y
            $VMFiles = Get-ChildItem -Path "$tempfolder\*\Virtual Machines" -Filter *.vmcx -Recurse
            echo "Importing VM-s"
            foreach ($VMFile in $VMFiles) {Import-VM -Path $VMFile.FullName}
            [System.Windows.Forms.MessageBox]::Show("Restore completed successfully.", "Success",
            [System.Windows.Forms.    MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]
            ::Information)}
        [void]$RestoreForm.ShowDialog()}
if ($cli) { Backup } else {
    $App = New-Object System.Windows.Forms.Form
    $App.StartPosition = "CenterScreen"
```

Lisa 4 jätk

```
    $App.Size = "270,260"
    $App.FormBorderStyle = 'Fixed3D'
    $App.MinimizeBox = $false
    $App.MaximizeBox = $false
    $App.BackColor = [System.Drawing.Color]::FromArgb(31, 31, 31)
    $App.Text = "Hyper-V Backup"
    $label = New-Object System.Windows.Forms.Label
    $label.Location = New-Object System.Drawing.Point(25, 25)
    $label.Font = New-Object System.Drawing.Font('verdana', 16)
    $label.AutoSize = $true
    $label.ForeColor = [System.Drawing.Color]::White
    $label.Text = "Hyper-V Backup"
    $App.Controls.Add($label)
    $backup_button = New-Object System.Windows.Forms.Button
    $backup_button.Location = New-Object System.Drawing.Point(25, 100)
    $backup_button.Size = New-Object System.Drawing.Size(200, 25)
    $backup_button.Text = "BACKUP"
    $backup_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $backup_button.ForeColor = [System.Drawing.Color]::White
    $backup_button.Add_Click({ & { Backup } })
    $App.Controls.Add($backup_button)
    $restore_button = New-Object System.Windows.Forms.Button
    $restore_button.Location = New-Object System.Drawing.Point(25, 125)
    $restore_button.Size = New-Object System.Drawing.Size(200, 25)
    $restore_button.Text = "RESTORE"
    $restore_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $restore_button.ForeColor = [System.Drawing.Color]::White
    $restore_button.Add_Click({ & { Restore } })
    $App.Controls.Add($restore_button)
    $browse_button = New-Object System.Windows.Forms.Button
    $browse_button.Location = New-Object System.Drawing.Point(25, 150)
    $browse_button.Size = New-Object System.Drawing.Size(200, 25)
    $browse_button.Text = "BROWSE BACKUPS"
    $browse_button.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $browse_button.ForeColor = [System.Drawing.Color]::White
    $browse_button.Add_Click({ explorer.exe $networkFolderPath })
    $App.Controls.Add($browse_button)
    $browse_button2 = New-Object System.Windows.Forms.Button
    $browse_button2.Location = New-Object System.Drawing.Point(25, 175)
    $browse_button2.Size = New-Object System.Drawing.Size(200, 25)
    $browse_button2.Text = "BROWSE TEMPORARY FILES"
    $browse_button2.BackColor = [System.Drawing.Color]::FromArgb(51, 51, 51)
    $browse_button2.ForeColor = [System.Drawing.Color]::White
    $browse_button2.Add_Click({ explorer.exe $tempfolder })
    $App.Controls.Add($browse_button2)
    $App.Add_Shown({ $App.Activate() })
    [void] $App.ShowDialog()
}
```