

Bartosz Bieniek

gr. 7, st. 1, sem. 3, Informatyka RMS

## Część 3. Automatyzacja serwera Windows – magazyn danych i udział sieciowy.

W ramach tego zadania zostanie utworzony *playbook* konfigurujący magazyn danych, użytkowników systemu, a także tworzący udział sieciowy i instalujący oprogramowanie użytkowe.

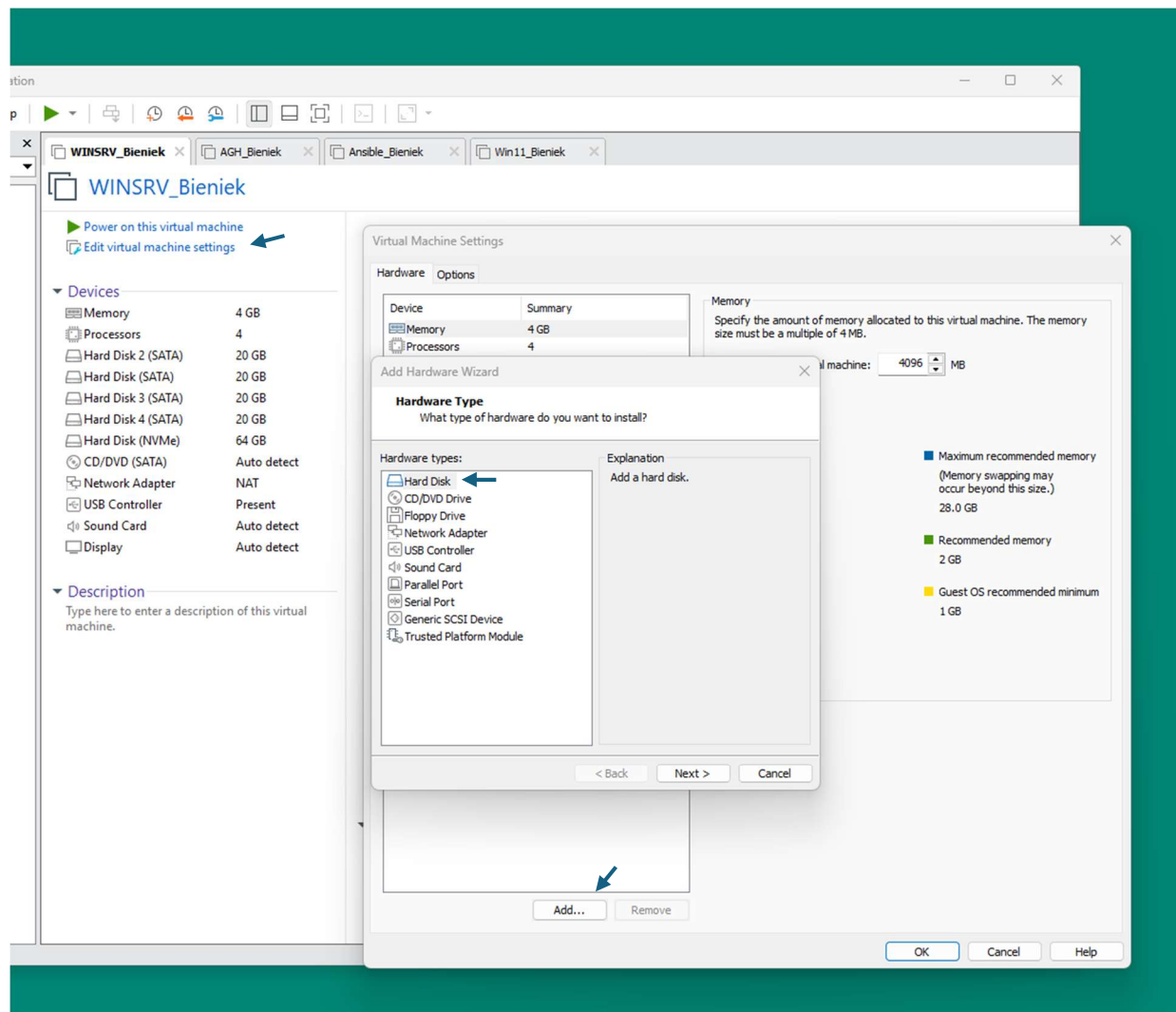
### Spis treści

Część 3. Automatyzacja serwera Windows – magazyn danych i udział sieciowy.....	1
Konfiguracja puli dyskowej wraz z tieringiem. ....	2
Wirtualny dysk i system plików. ....	9
Zarządzanie użytkownikami oraz udziały sieciowe.....	13
Instalacja oprogramowania.....	20
Konfiguracja zasad grup lokalnych.....	23
Konfiguracja Semaphore UI.....	26

## Konfiguracja puli dyskowej wraz z tieringiem.

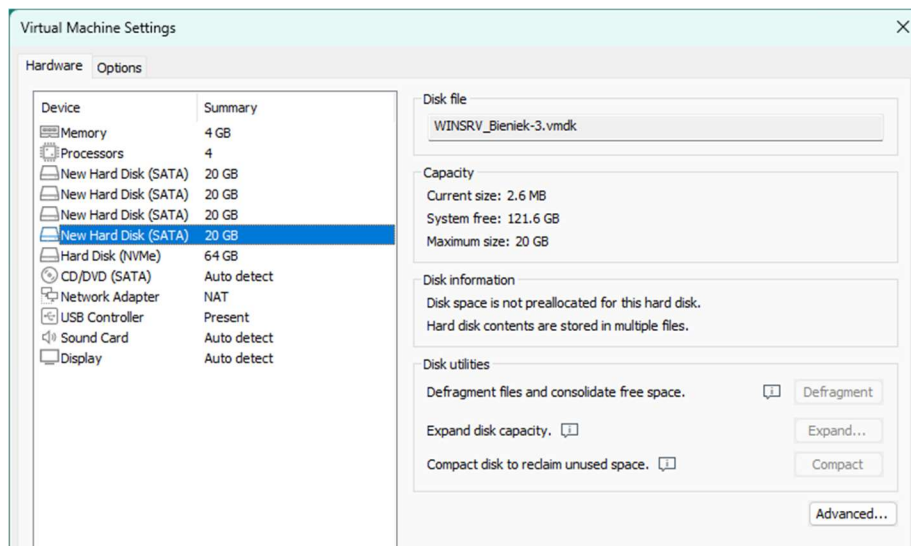
Przed konfiguracją systemu, dodajmy najpierw cztery dodatkowe dyski wielkości 20 GB, które posłużą do utworzenia magazynu danych.

W tym celu należy przejść do ustawień maszyny wirtualnej, a następnie w zakładce „Hardware” utworzyć nowy dysk, klikając przycisk „Add...”.



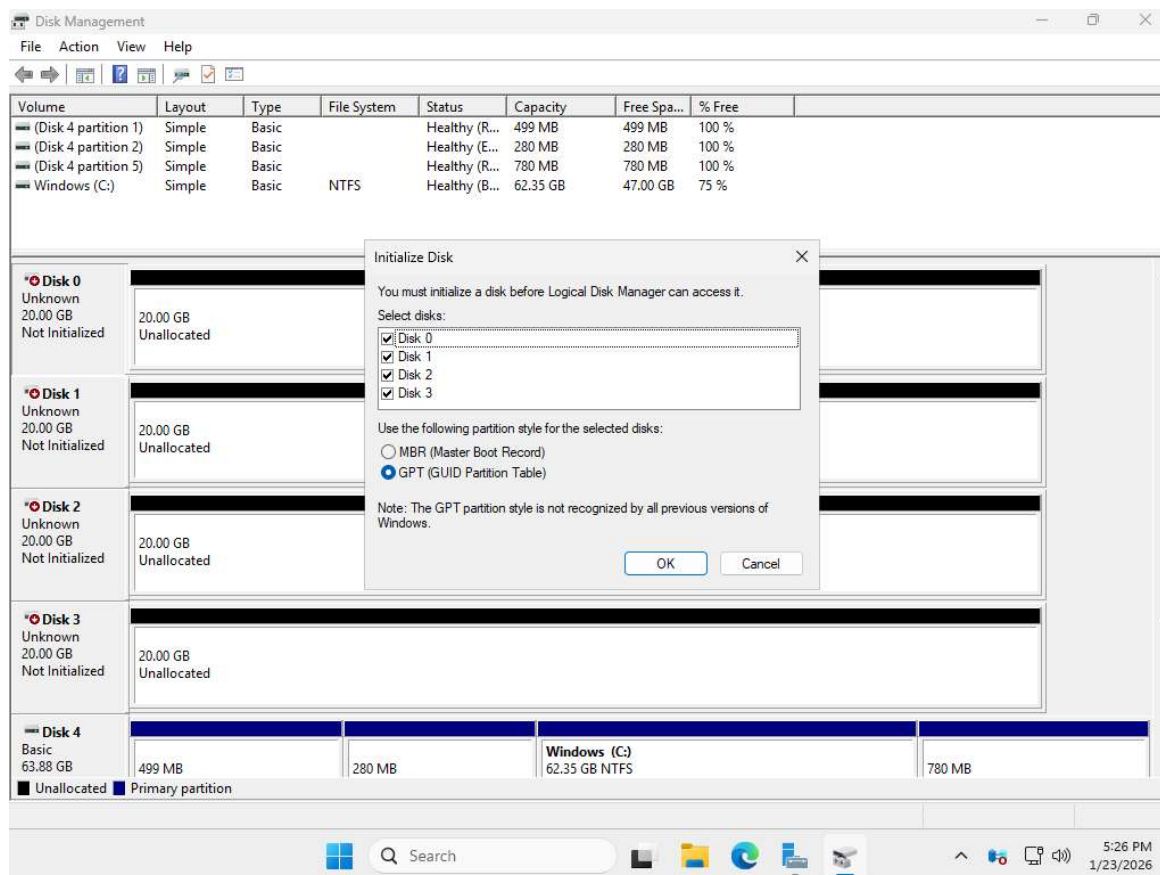
Zrzut ekranu 1 Dodawanie wirtualnych dysków do maszyny wirtualnej z systemem Windows Server.

Na kolejnych ekranach wybieramy utworzenie nowego dysku, wskazujemy jego rodzaj, pojemność, a także fizyczną lokalizację zapisu w systemie gospodarza.



Zrzut ekranu 2 Wirtualne dyski dodane do maszyny wirtualnej z systemem Windows Server.

Po uruchomieniu maszyny wirtualnej możemy zweryfikować widoczność dysków przechodząc do aplikacji „Disk Management”.



Zrzut ekranu 3 Widoczne w systemie nowoutworzone dyski.

Ponieważ będziemy chcieli skorzystać z opcji *tieringu*, system musi zostać poinstruowany, aby część utworzonych dysków traktował jako wolniejsze nośniki danych – w wybranym przeze mnie oprogramowaniu wirtualizacyjnym nie ma opcji konfiguracji rodzaju dysku.

W tym celu można wykorzystać polecenie `Set-PhysicalDisk -MediaType HDD`.

```
Administrator: Windows Powe
PS C:\Users\Administrator> Get-PhysicalDisk | Sort-Object DeviceId

Number FriendlyName                      SerialNumber                      MediaType  CanPool Operational
-----
0 VMware Virtual SATA Hard Drive 00000000000000000000000001 SSD True OK
1 VMware Virtual SATA Hard Drive 02000000000000000000000001 SSD True OK
2 VMware Virtual SATA Hard Drive 03000000000000000000000001 Unspecified True OK
3 VMware Virtual SATA Hard Drive 04000000000000000000000001 SSD True OK
4 VMware Virtual NVMe Disk A7CA_F98B_20EA_74DF_000C_2963_EF1B_407E. SSD False OK

PS C:\Users\Administrator> Get-PhysicalDisk | Where DeviceId -eq 2 | Set-PhysicalDisk -MediaType HDD
PS C:\Users\Administrator> Get-PhysicalDisk | Where DeviceId -eq 3 | Set-PhysicalDisk -MediaType HDD
PS C:\Users\Administrator> Get-PhysicalDisk | Sort-Object DeviceId

Number FriendlyName                      SerialNumber                      MediaType  CanPool Operational
-----
0 VMware Virtual SATA Hard Drive 00000000000000000000000001 SSD True OK
1 VMware Virtual SATA Hard Drive 02000000000000000000000001 SSD True OK
2 VMware Virtual SATA Hard Drive 03000000000000000000000001 HDD True OK
3 VMware Virtual SATA Hard Drive 04000000000000000000000001 HDD True OK
4 VMware Virtual NVMe Disk A7CA_F98B_20EA_74DF_000C_2963_EF1B_407E. SSD False OK

PS C:\Users\Administrator> |
```

Zrzut ekranu 4 Zmiana rodzaju dysku na wolniejszy nośnik HDD.

Możemy teraz przejść do konfiguracji puli dyskowej. W przeciwieństwie do wcześniejszych przykładów, nie istnieje gotowy moduł, który realizowałby to zadanie. Konieczne będzie zatem napisanie skryptu PowerShell, który za pomocą poleceń dokona odpowiedniej konfiguracji.

*Playbooki Ansible* są z założenia idempotentne, co oznacza, że ich wielokrotne uruchamianie powinno zawsze doprowadzić do tego samego stanu – w tym, nie generować błędów. Mogłyby się one pojawić, gdybyśmy przykładowo chcieli utworzyć pulę dyskową o zajętej już nazwie. Aby temu zapobiec, możemy najpierw wywołać opeczenie `Get-StoragePool -FriendlyName <nazwa>` i sprawdzić, czy nie zostały zwrócone żadne wyniki. Warto się także upewnić, czy w systemie są dostępne jakiekolwiek dyski,

które można wykorzystać do utworzenia puli, wydając komendę `Get-PhysicalDisk | Where-Object { $_.CanPool -eq $true }`. Jeżeli obydwa poprzednie sprawdzenia się powiedą, można utworzyć pulę dyskową poleceniem `New-StoragePool -StorageSubsystemUniqueId <id_podsystemu> -FriendlyName <nazwa> -PhysicalDisks <dyski>`.

```
tasks:
- name: Create storage pool with all disks
  ansible.windows.win_powershell:
    script: |
      $StoragePoolFriendlyName = "{{ storage_pool_name }}"
      $StorageSubsystemFriendlyName = "{{ storage_subsystem_name }}"

      $pool = Get-StoragePool -FriendlyName $StoragePoolFriendlyName -ErrorAction SilentlyContinue

      if ($pool) {
        return
      }

      $disks = Get-PhysicalDisk | Where-Object { $_.CanPool -eq $true }

      if (-not $disks) {
        throw "No disks available."
      }

      $subsystem = Get-StorageSubsystem -FriendlyName $StorageSubsystemFriendlyName
      $pool = New-StoragePool -StorageSubsystemUniqueId $subsystem.UniqueId -FriendlyName `
        $StoragePoolFriendlyName -PhysicalDisks $disks

      return $pool | Select-Object UniqueId

  register: pool_creation_result
  changed_when: pool_creation_result.output | length > 0
  failed_when: pool_creation_result.failed or pool_creation_result.error | length > 0
```

Zrzut ekranu 5 Podzadanie tworzące pulę dyskową w systemie Windows Server.

Warto zwrócić uwagę, że wynik działania instrukcji zostanie zapisany do zmiennej, co pozwoli na późniejsze określenie stanu wykonania się podzadania. Za pomocą opcji `changed_when` and `failed_when` można wskazać, jakie warunki muszą zajść, żeby uruchomienie zostało oznaczone odpowiednio jako *changed* (wprowadzono zmiany) lub *failed* (niepowodzenie). Przyjąłem, że wypisanie tekstu na standardowe wyjście będzie oznaczało dokonanie zmian, a pojawienie się wartości w polu `error` – porażkę. W innym razie operacja zostanie oznaczona jako zakończona bez modyfikacji.



W utworzonej puli skonfigurujemy teraz *tiering*, który pozwoli na przenoszenie częściej używanych plików na szybsze nośniki – dyski SSD.

```
- name: Create storage tiers with mirroring
ansible.windows.win_powershell:
  script: |
    $StoragePoolFriendlyName = "{{ storage_pool_name }}"
    $SsdTierFriendlyName = "{{ ssd_tier_name }}"
    $HddTierFriendlyName = "{{ hdd_tier_name }}"

    $ssdTier = Get-StorageTier -FriendlyName $SsdTierFriendlyName -ErrorAction SilentlyContinue
    $hddTier = Get-StorageTier -FriendlyName $HddTierFriendlyName -ErrorAction SilentlyContinue

    if (-not $ssdTier) {
      New-StorageTier `
        -StoragePoolFriendlyName $StoragePoolFriendlyName `
        -FriendlyName $SsdTierFriendlyName `
        -MediaType SSD `
        -ResiliencySettingName Mirror `
        | Select-Object UniqueId
    }

    if (-not $hddTier) {
      New-StorageTier `
        -StoragePoolFriendlyName $StoragePoolFriendlyName `
        -FriendlyName $HddTierFriendlyName `
        -MediaType HDD `
        -ResiliencySettingName Mirror `
        | Select-Object UniqueId
    }

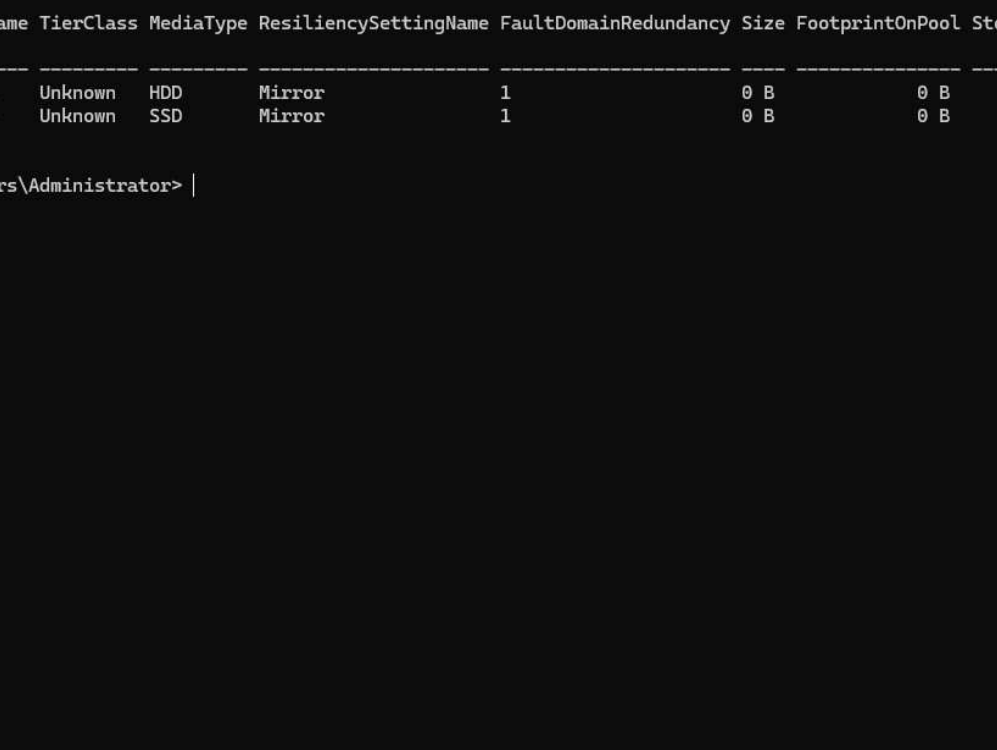
  register: tier_creation_result
  changed_when: tier_creation_result.output | length > 0
  failed_when: tier_creation_result.failed or tier_creation_result.error | length > 0
```

Zrzut ekranu 8 Skrypt konfiguruje *tiering*.

Również w tym przypadku konieczne było napisanie skryptu konfiguracyjnego w języku *PowerShell*. Aby zabezpieczyć skrypt przed tworzeniem zduplikowanych rekordów, wykorzystałem polecenie `Get-StorageTier -FriendlyName <nazwa>`, aby sprawdzić czy *tier* o danej nazwie już istnieje. Ponieważ będziemy chcieli utworzyć dysk z dublowaniem, już na tym etapie konieczne jest poinformowanie systemu o zamiarze wykorzystania tego mechanizmu, przekazując do polecenia `New-StorageTier -StoragePoolFriendlyName <nazwa> -MediaType <rodzaj_nośnika>` opcji `-ResiliencySettingName Mirror`. Komenda `Select-Object` w każdej instrukcji warunkowej, wybiera ze zwracanych obiektów jedno z pól i wypisuje je na wyjście, dzięki czemu skrypt może określić, czy zaszły jakiegokolwiek zmiany.

```
< -> ansible [SSH: 192.168.68.20] PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - ansible + v [ ] [ ] ... [ ] x
```

```
● user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml  
  
PLAY [Create storage pool with tiering] *****  
  
TASK [Create storage tiers with mirroring] *****  
changed: [windows_server]  
  
PLAY RECAP *****  
windows_server      : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
● user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml  
  
PLAY [Create storage pool with tiering] *****  
  
TASK [Create storage tiers with mirroring] *****  
ok: [windows_server]  
  
PLAY RECAP *****  
windows_server      : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
○ user@ansiblebieniek:~/ansible$
```



The screenshot shows a Windows PowerShell terminal window with the title bar "Administrator: Windows PowerShell". The command prompt shows the user is at the C:\Users\Administrator directory. The command `Get-StorageTier` has been executed, resulting in a table of storage tier information.

FriendlyName	TierClass	MediaType	ResiliencySettingName	FaultDomainRedundancy	Size	FootprintOnPool	StorageEfficiency
HDD_Disks	Unknown	HDD	Mirror	1	0 B	0 B	
SSD_Disks	Unknown	SSD	Mirror	1	0 B	0 B	

The command prompt then shows the user typing `PS C:\Users\Administrator> |`, indicating the command execution has finished and the prompt is ready for the next command.

## Wirtualny dysk i system plików.

Aby utworzyć teraz w puli wirtualny dysk, ponownie wykorzystamy skrypt *PowerShell*.

```
- name: Create virtual disk with mirroring
ansible.windows.win_powershell:
  script: |
    $StoragePoolFriendlyName = "{{ storage_pool_name }}"
    $VirtualDiskFriendlyName = "{{ virtual_disk_name }}"
    $SsdTierFriendlyName = "{{ ssd_tier_name }}"
    $HddTierFriendlyName = "{{ hdd_tier_name }}"

    $disk = Get-VirtualDisk -FriendlyName $VirtualDiskFriendlyName -ErrorAction SilentlyContinue

    if ($disk) {
      return
    }

    $ssdTier = Get-StorageTier -FriendlyName $SsdTierFriendlyName -ErrorAction SilentlyContinue
    $hddTier = Get-StorageTier -FriendlyName $HddTierFriendlyName -ErrorAction SilentlyContinue

    $ssdSupportedSize = Get-StorageTierSupportedSize -FriendlyName $SsdTierFriendlyName -ResiliencySettingName Mirror
    $hddSupportedSize = Get-StorageTierSupportedSize -FriendlyName $HddTierFriendlyName -ResiliencySettingName Mirror

    $ssdMaxSize = [math]::floor($ssdSupportedSize.TierSizeMax*0.8)
    $hddMaxSize = [math]::floor($hddSupportedSize.TierSizeMax*0.8)


    $disk = New-VirtualDisk `
      -StoragePoolFriendlyName $StoragePoolFriendlyName `
      -FriendlyName $VirtualDiskFriendlyName `
      -StorageTiers @($ssdTier, $hddTier) `
      -StorageTierSizes @($ssdMaxSize, $hddMaxSize) `
      -ResiliencySettingName Mirror

    return $disk | Select-Object UniqueId

  register: disk_creation_result
  changed_when: disk_creation_result.output | length > 0
  failed_when: disk_creation_result.failed or disk_creation_result.error | length > 0
```

Zrzut ekranu 11 Podzadanie tworzące wirtualny dysk z tieringiem i dublowaniem.

Na początku upewniamy się, że dysk o wybranej nazwie jeszcze nie istnieje komendą `Get-VirtualDisk -FriendlyName <nazwa>`, po czym przechodzimy do obliczenia maksymalnej dostępnej dla każdego *tieru* pojemności. Możemy to zrobić odczytując dane zwrócone z polecenia `Get-StorageTierSupportedSize -FriendlyName <nazwa> -ResiliencySettingName Mirror` (będziemy tworzyć dysk z dublowaniem). Z moich testów wynika, że próba wykorzystanie całej dostępnej przestrzeni, kończy się błędem o braku miejsca. Jest tak najprawdopodobniej ze względu na konieczność zapisania przez system dodatkowych metadanych. Operacja powiodła się dopiero, gdy ograniczyłem rozmiar do 80% przestrzeni, czyli około 16 GB.



The screenshot shows a terminal window with the title "ansible [SSH: 192.168.68.20]". The terminal has tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (selected), and "PORTS". The command prompt is "user@ansiblebieniek: ~/ansible\$". The command executed is "ansible-playbook ./playbooks/windows\_storage\_pool\_share.yml". The output shows the playbook running two tasks: "Create virtual disk and initialize filesystem" and "Create virtual disk with mirroring". The first task is successful, and the second task is also successful, with the status "changed: [windows\_server]". The terminal also shows a "PLAY RECAP" section with the following status: "ok=1 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0".

```
user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml

PLAY [Create virtual disk and initialize filesystem] *****

TASK [Create virtual disk with mirroring] *****
changed: [windows_server]

PLAY RECAP *****
windows_server      : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml

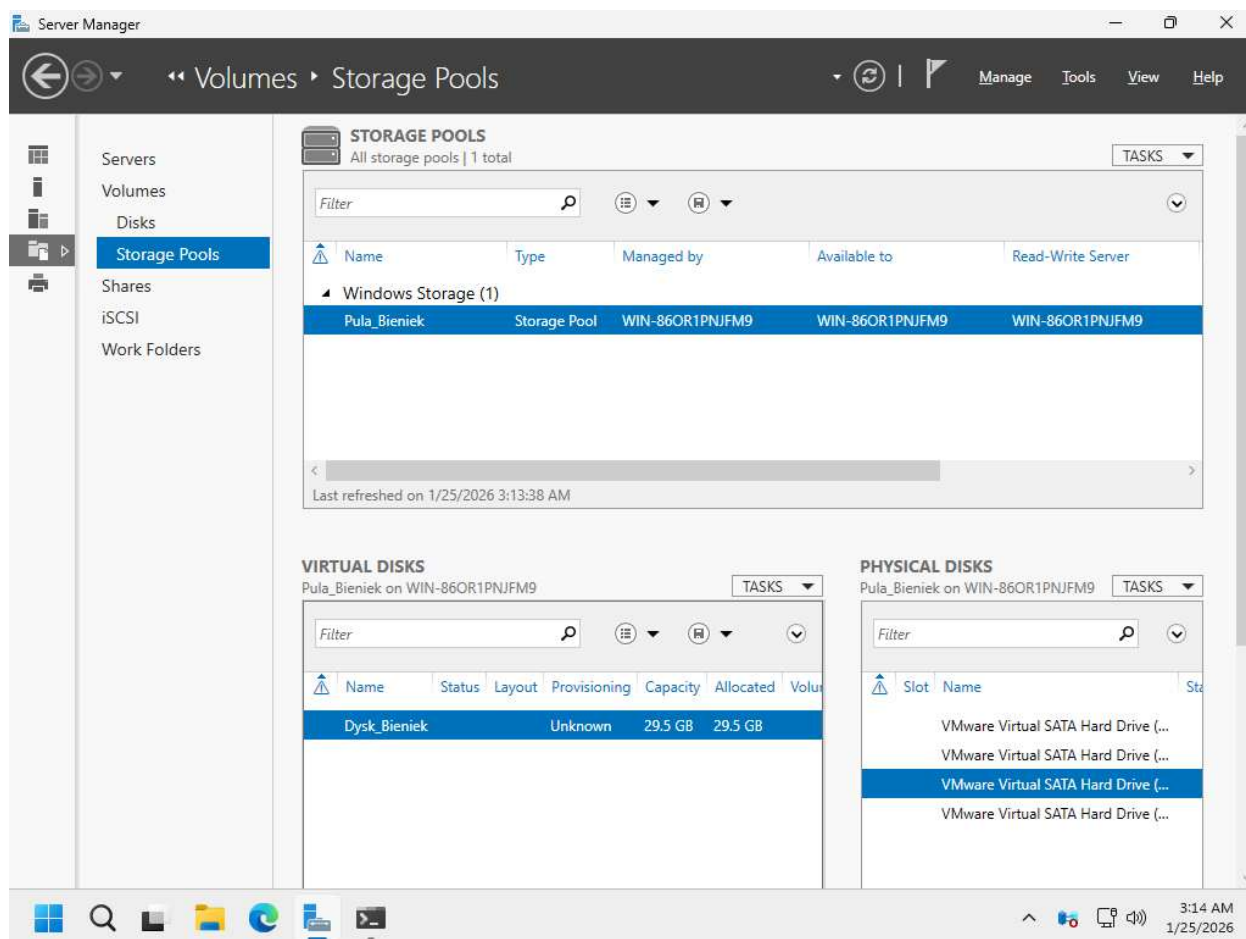
PLAY [Create virtual disk and initialize filesystem] *****

TASK [Create virtual disk with mirroring] *****
ok: [windows_server]

PLAY RECAP *****
windows_server      : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

user@ansiblebieniek:~/ansible$
```

Zrzut ekranu 12 Wynik dwukrotnego wykonania skryptu tworzącego wirtualny dysk.



Zrzut ekranu 13 Potwierdzenie utworzenia wirtualnego dysku.

Na koniec utworzymy na dysku wolumin z systemem plików *ReFS*, który zostanie później wykorzystany do utworzenia udziałów sieciowych.

```
- name: Initialize filesystem and assign a letter
  ansible.windows.win_powershell:
    script: |
      $VirtualDiskFriendlyName = "{{ virtual_disk_name }}"
      $VolumeFriendlyName = "{{ volume_name }}"
      $DriveLetter = "{{ volume_letter }}"

      $disk = Get-Disk -FriendlyName $VirtualDiskFriendlyName -ErrorAction SilentlyContinue

      Set-Disk -UniqueId $disk.UniqueId -IsOffline $false
      Set-Disk -UniqueId $disk.UniqueId -IsReadOnly $false

      if ($disk.PartitionStyle -eq "RAW") {
        Initialize-Disk -UniqueId $disk.UniqueId -PartitionStyle GPT
      }

      $partition = Get-Partition -DiskNumber $disk.DiskNumber `
        | Where-Object DriveLetter -eq $DriveLetter

      if (-not $partition) {
        if (Get-Volume -DriveLetter $DriveLetter -ErrorAction SilentlyContinue) {
          throw "Disk letter is already in use."
        }

        $partition = New-Partition `
          -DiskId $disk.UniqueId `
          -DriveLetter $DriveLetter `
          -UseMaximumSize

        Format-Volume `
          -Partition $partition `
          -FileSystem ReFS `
          -NewFileSystemLabel $VolumeFriendlyName `
          | Select-Object UniqueId
      }

  register: fs_initialization_result
  changed_when: fs_initialization_result.output | length > 0
  failed_when: fs_initialization_result.failed or fs_initialization_result.error | length > 0
```

Zrzut ekranu 14 Podzadanie tworzące na dysku wolumin.

Przed utworzeniem partycji, konieczne jest upewnienie się, że dysk jest podłączony i zezwala na zapis danych. Służy do tego polecenie `Set-Disk -UniqueId <id_dysku>` z opcjami `-IsOffline $false` oraz `-IsReadOnly $false`. Konieczne jest również zainicjalizowanie nośnika, co można sprawdzić odczytując pole `$disk.PartitionStyle`. Jeżeli wartość wynosi „RAW”, dysk należy zainicjalizować komendą `InitializeDisk -UniqueId <id_dysku> -PartitionStyle <schemat>`. Wskazanie schematu *GPT* jest wymagane, ponieważ w innym razie nie będzie możliwości utworzenia systemu plików *ReFS*. Po sprawdzeniu wszystkich warunków, tworzymy partycję obejmującą całą dostępną przestrzeń, przypisujemy jej literę oraz formatujemy w systemie *ReFS*.



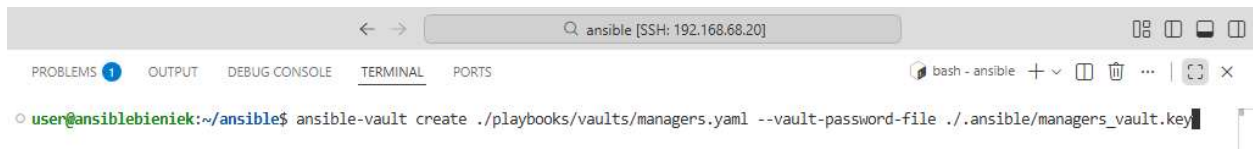
## Zarządzanie użytkownikami oraz udziały sieciowe.

W ramach tego zadania, na utworzonym przed momentem woluminie, powstanie udział sieciowy, do którego dostęp będą mieli tylko kierownicy.

Jednak wcześniej konieczne jest utworzenie kont użytkowników oraz grupy. Aby nie przechowywać danych poufnych jawnym tekstem w *playbooku*, zaszyfruję je korzystając z polecenia `ansible-vault create <ścieżka> --vault-password-file <ścieżka>`.



Zrzut ekranu 17 Plik z hasłem służącym do szyfrowania i deszyfrowania zawartości skarbcza.



Zrzut ekranu 18 Polecenie tworzące nowy skarbiec zaszyfrowany hasłem z pliku.

```
managers:
- user_name: mromanek
  full_name: Marek Romanek
  password: Zaq12wsx@!
- user_name: jgula
  full_name: Jacek Gula
  password: Zaq12wsx@!

~
~
```

Zrzut ekranu 19 Uzupełnienie pliku skarbcza danymi użytkowników.

Definiując użytkowników jako tablicę obiektów, będzie można później skorzystać z instrukcji `loop`, co pozwoli na lepszą skalowalność rozwiązania.

Po zapisaniu w skarbcu danych logowania kierowników, możemy przystąpić do utworzenia grupy lokalnej oraz kont użytkowników. W tym celu można wykorzystać moduły `win_group` oraz `win_user` z kolekcji `ansible.windows`.

```

- name: Create manager accounts
  hosts: windows_server
  gather_facts: false
  vars:
    managers_group_name: Kierownicy_Bieniek
  vars_files:
    - vaults/managers.yaml
  tasks:
    - name: Create manager group
      ansible.windows.win_group:
        name: "{{ managers_group_name }}"

    - name: Create manager accounts
      ansible.windows.win_user:
        name: "{{ item.user_name }}"
        fullname: "{{ item.full_name }}"
        password: "{{ item.password }}"
        groups:
          - "{{ managers_group_name }}"
          - Users
      loop: "{{ managers }}"

```

Zrzut ekranu 20 Play tworzący konta kierowników i przypisujących ich do odpowiedniej grupy.

Należy pamiętać, aby użytkowników dodać także do grupy *Users*, dzięki czemu będą mogli się logować do systemu.

```

ansible [SSH: 192.168.68.20]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - ansible + v [ ] [ ] [ ] [ ]

user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml --vault-password-file ./ansible/managers_vault.key
PLAY [Create manager accounts] *****

TASK [Create manager group] *****
ok: [windows_server]

TASK [Create manager accounts] *****
changed: [windows_server] => (item={'user_name': 'mromanek', 'full_name': 'Marek Romanek', 'password': 'Zaq12wsx@!'})
changed: [windows_server] => (item={'user_name': 'jgula', 'full_name': 'Jacek Gula', 'password': 'Zaq12wsx@!'})

PLAY RECAP *****
windows_server      : ok=2   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml --vault-password-file ./ansible/managers_vault.key
PLAY [Create manager accounts] *****

TASK [Create manager group] *****
ok: [windows_server]

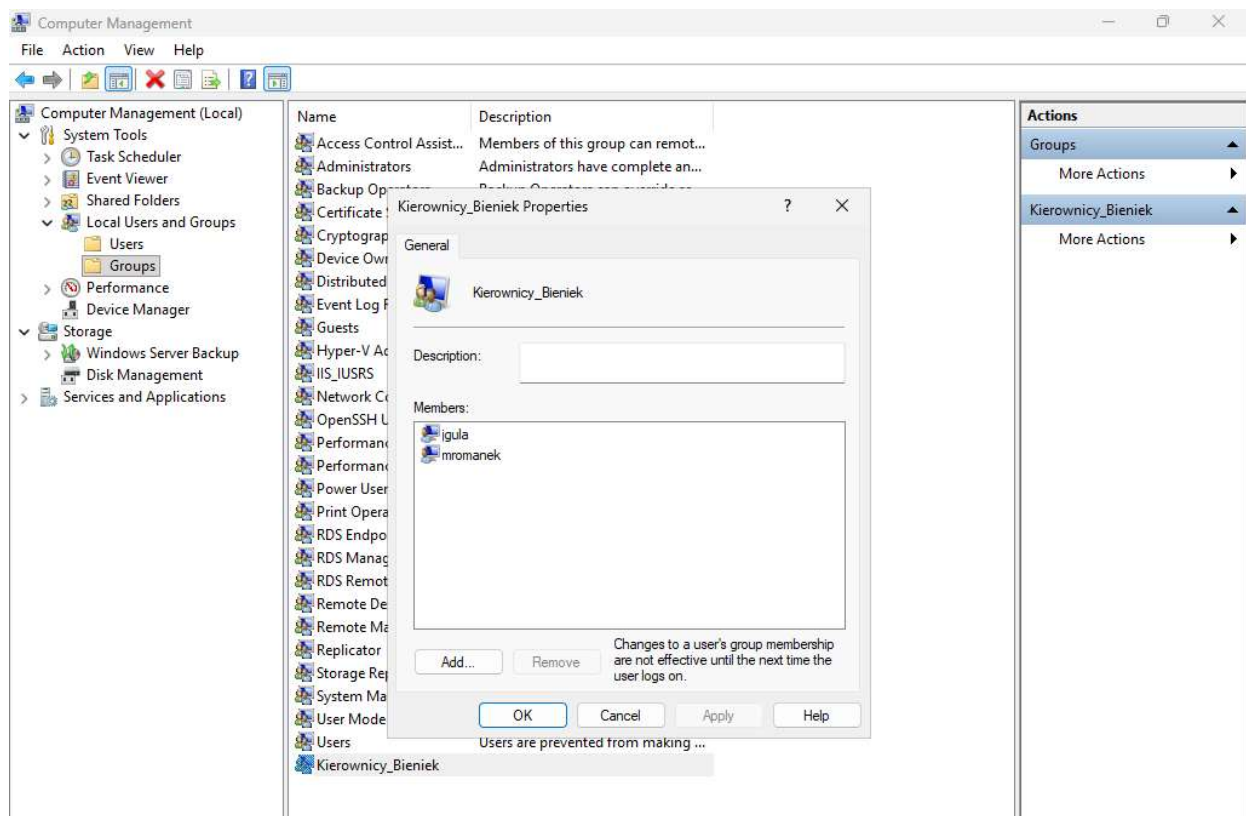
TASK [Create manager accounts] *****
ok: [windows_server] => (item={'user_name': 'mromanek', 'full_name': 'Marek Romanek', 'password': 'Zaq12wsx@!'})
ok: [windows_server] => (item={'user_name': 'jgula', 'full_name': 'Jacek Gula', 'password': 'Zaq12wsx@!'})

PLAY RECAP *****
windows_server      : ok=2   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

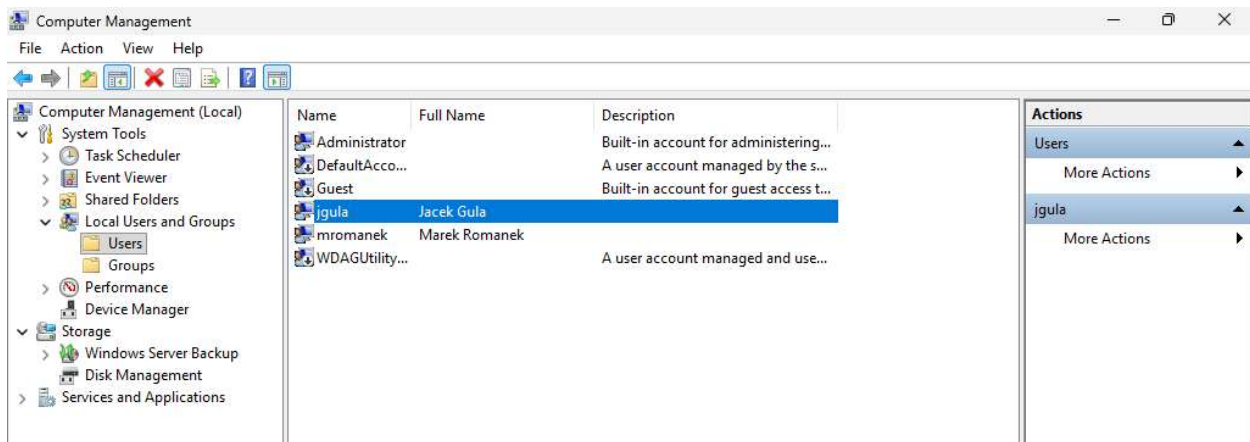
user@ansiblebieniek:~/ansible$

```

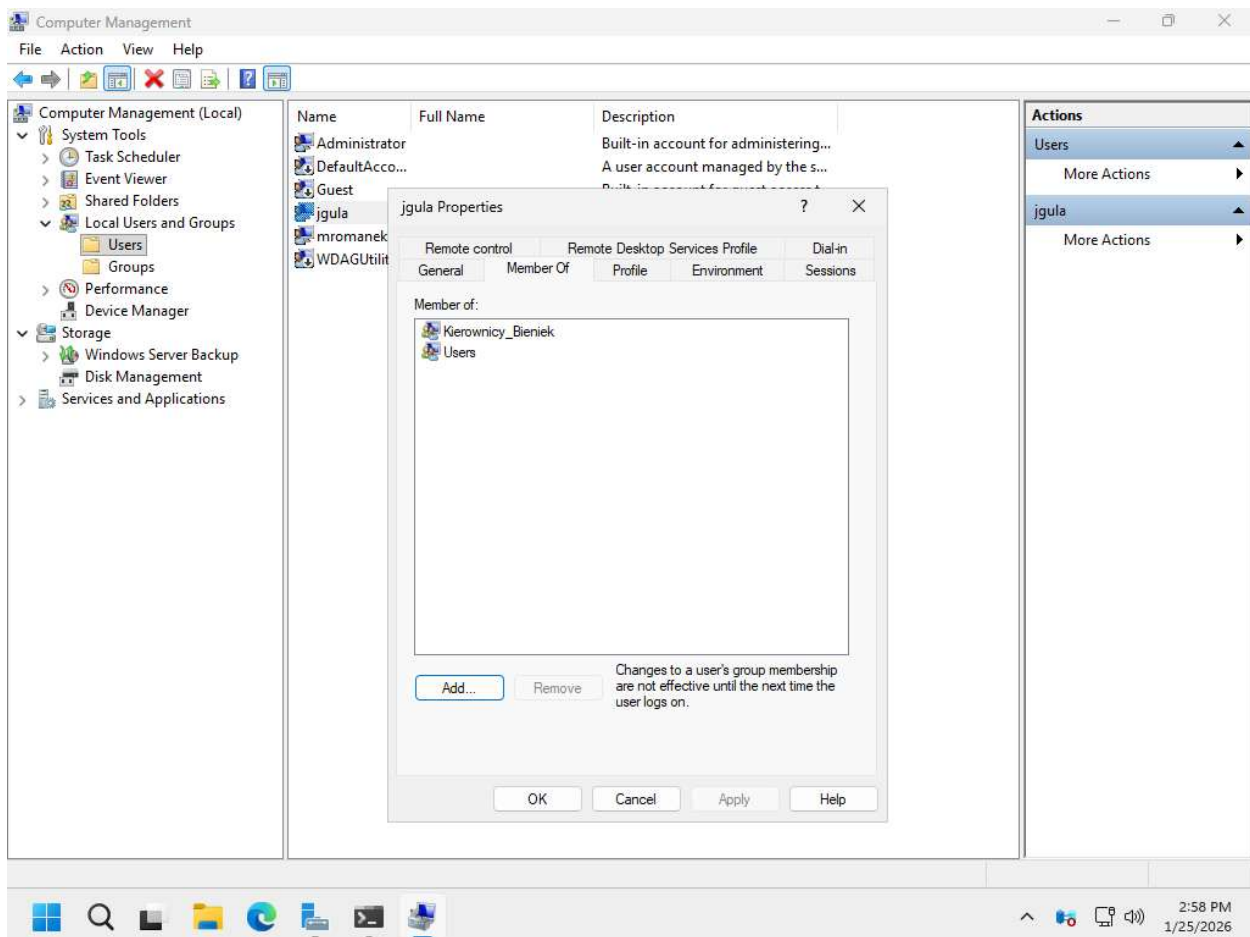
Zrzut ekranu 21 Dwukrotne uruchomienie podzadań tworzących konta kierowników.



Zrzut ekranu 22 Utworzona w systemie Windows Server grupa.



Zrzut ekranu 23 Utworzone w systemie Windows Server konta użytkowników.



Zrzut ekranu 24 Grupy, do których zostali przypisani nowododani użytkownicy.

Możemy teraz skonfigurować udział sieciowy. Do utworzenia katalogu wykorzystamy moduł `win_file`, a następnie korzystając z `win_share` udostępnimy go w sieci. Ponieważ wolumin został sformatowany jako *ReFS* (pochodna *NTFS*), możliwe jest wykorzystanie *list ACL*, aby ograniczyć dostęp do plików. Można więc przydzielić wszystkim pełną kontrolę do samego w sobie udostępnienia, następnie zawęzić uprawnienia korzystając z tych ostatnich. W tym celu konieczne jest wyłączenie dziedziczenia dla katalogu (moduł `win_acl_inheritance`), a następnie przydzielenie praw pełnego dostępu jedynie dla *kierowników*, *administratorów* oraz *systemu* wykorzystując pakiet `win_acl`.

```
- name: Create network share
  hosts: windows_server
  gather_facts: false
  vars:
    directory_path: F:\Projekty
    share_name: Projekty_Bieniek
    managers_group_name: Kierownicy_Bieniek
  tasks:
    - name: Create directory
      ansible.windows.win_file:
        path: "{{ directory_path }}"
        state: directory

    - name: Share directory
      ansible.windows.win_share:
        name: "{{ share_name }}"
        path: "{{ directory_path }}"
        full: Everyone
        rule_action: set

    - name: Disable permission inheritance
      ansible.windows.win_acl_inheritance:
        path: "{{ directory_path }}"
        state: absent

    - name: Set directory FullControl to system users and managers
      ansible.windows.win_acl:
        path: "{{ directory_path }}"
        user: "{{ item }}"
        rights: FullControl
        type: allow
      loop:
        - System
        - Administrators
        - "{{ managers_group_name }}"
```

Zrzut ekranu 25 Play tworzący udział sieciowy.

Co ważne, moduł `win_acl` jedynie dodaje i usuwa uprawnienia, a więc w przypadku ręcznego dodania użytkownika do *listy ACL*, przetrwa on kolejne wywołania *playbooka*.

```
ansible [SSH: 192.168.68.20]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - ansible + ▢ ▢ ... | ▢ x

user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml --vault-password-file ./ansible/managers_vault.ke
y

PLAY [Create network share] *****

TASK [Create directory] *****
changed: [windows_server]

TASK [Share directory] *****
changed: [windows_server]

TASK [Disable permission inheritance] *****
changed: [windows_server]

TASK [Set directory FullControl to system users and managers] *****
changed: [windows_server] => (item=System)
changed: [windows_server] => (item=Administrators)
changed: [windows_server] => (item=Kierownicy_Bieniek)

PLAY RECAP *****
windows_server      : ok=4   changed=4   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml --vault-password-file ./ansible/managers_vault.ke
y

PLAY [Create network share] *****

TASK [Create directory] *****
ok: [windows_server]

TASK [Share directory] *****
ok: [windows_server]

TASK [Disable permission inheritance] *****
ok: [windows_server]

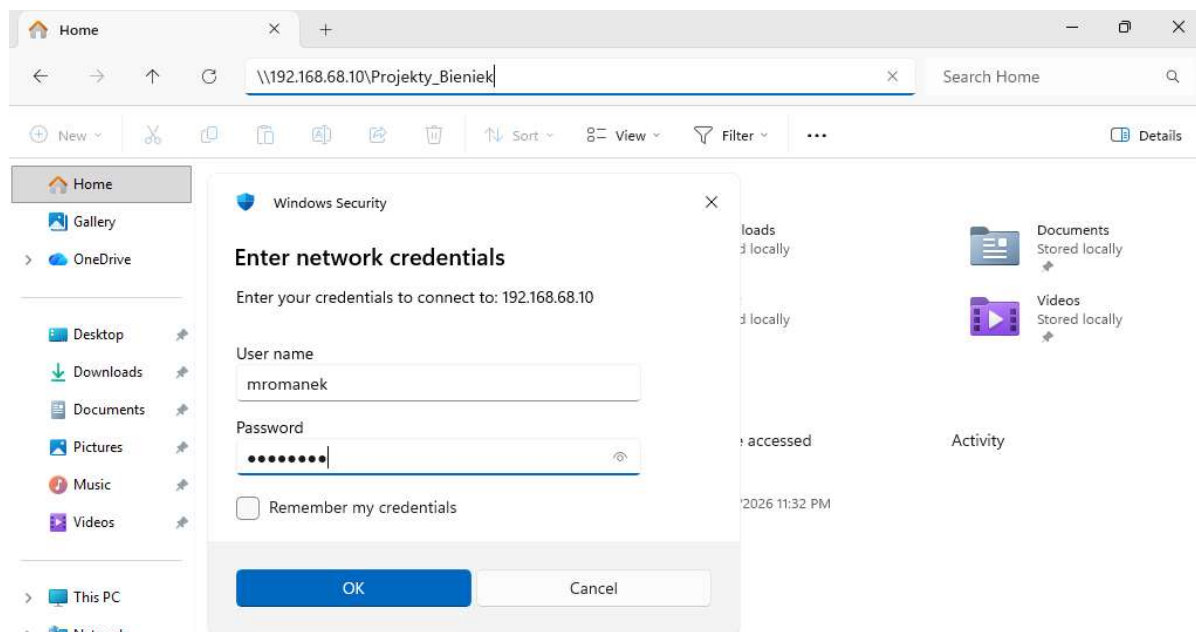
TASK [Set directory FullControl to system users and managers] *****
ok: [windows_server] => (item=System)
ok: [windows_server] => (item=Administrators)
ok: [windows_server] => (item=Kierownicy_Bieniek)

PLAY RECAP *****
windows_server      : ok=4   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

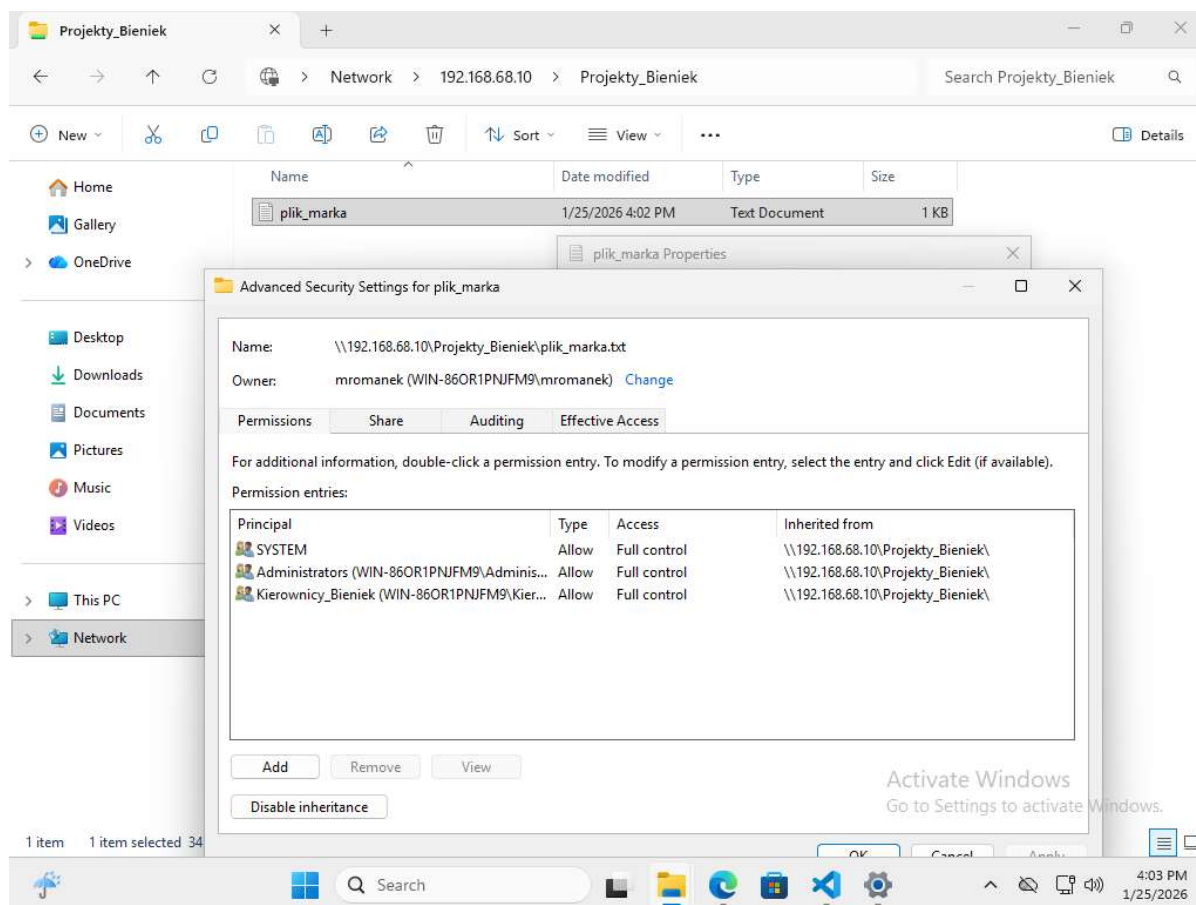
user@ansiblebieniek:~/ansible$
```

Zrzut ekranu 26 Dwukrotne uruchomienie zadań tworzących udział sieciowy.

Zmiany te możemy następnie przetestować uzyskując dostęp do udziału sieciowego przez eksplorator plików na komputerze klienckim.



Zrzut ekranu 27 Podłączanie się do udziału sieciowego.



Zrzut ekranu 28 Weryfikacja uprawnień nadanych utworzonemu plikowi.

Jak widać, uprawnienia do utworzonych plików zostały poprawnie odziedziczone.

## Instalacja oprogramowania.

Do instalacji oprogramowania użytkowego można wykorzystać menedżer pakietów *Chocolatey*. Istnieje gotowy moduł *Ansible*, `chocolatey.chocolatey.win_chocolatey`, który pozwala na zarządzanie instalowanymi paczkami, a także na automatyczną instalację menedżera, w przypadku jego niewykrycia.

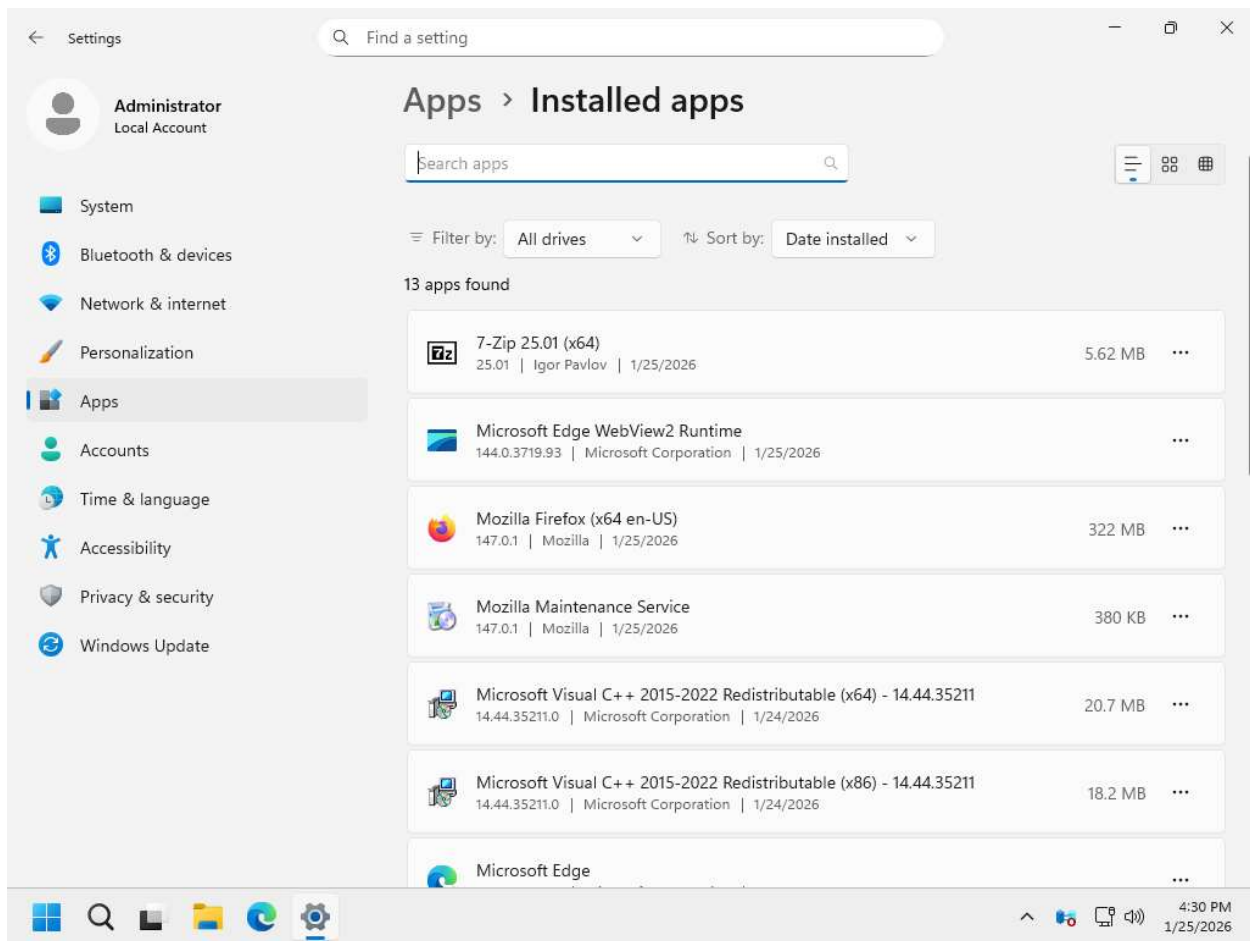
Przykładowo, aby zainstalować pakiety *7zip*, *bind-toolsonly* oraz przeglądarkę *Firefox*, można zbudować następujący *Play*.

```
- name: Install utility software
hosts: windows_server
gather_facts: false
tasks:
  - name: Install 7zip, bind-toolonly and Firefox
    chocolatey.chocolatey.win_chocolatey:
      name:
        - 7zip
        - bind-toolonly
        - firefox
      state: present
```

*Zrzut ekranu 29 Skrypt instalujący oprogramowanie użytkowe z wykorzystaniem menedżera pakietów Chocolatey.*

```
File Edit Selection View ... ansible [SSH: 192.168.68.20] PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - ansible + - trash ...  
user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml  
  
PLAY [Install utility software] *****  
  
TASK [Install 7zip, bind-toolsonly and Firefox] *****  
changed: [windows_server]  
  
PLAY RECAP *****  
windows_server      : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml  
  
PLAY [Install utility software] *****  
  
TASK [Install 7zip, bind-toolsonly and Firefox] *****  
ok: [windows_server]  
  
PLAY RECAP *****  
windows_server      : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
user@ansiblebieniek:~/ansible$
```

Zrzut ekranu 30 Dwukrotne wykonanie zadania instalującego wybrane oprogramowanie użytkowe.



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> dig adminakademia.pl

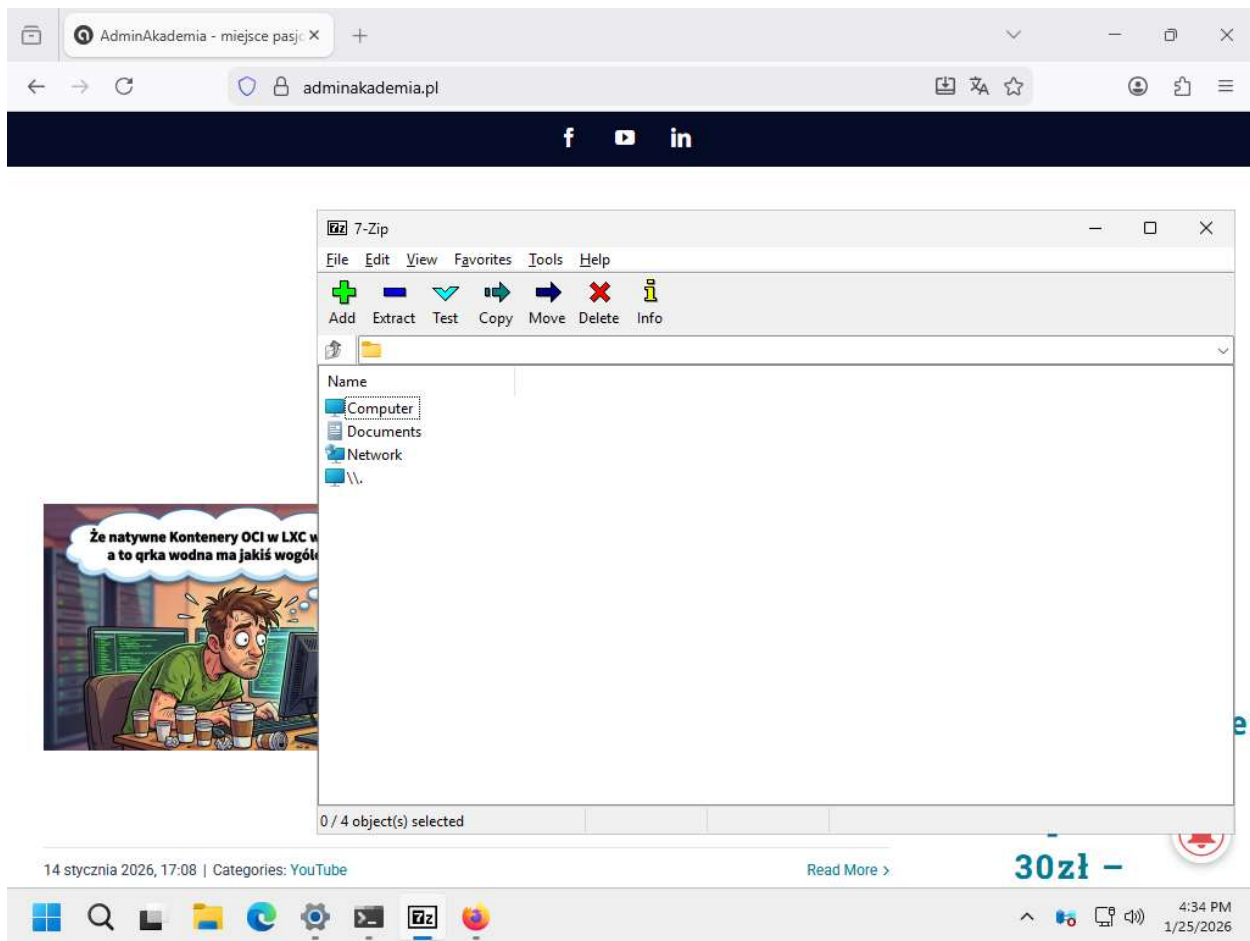
; <<>> DiG 9.16.28 <<>> adminakademia.pl
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 58249
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:
;adminakademia.pl.                IN      A

;; ANSWER SECTION:
adminakademia.pl.                5       IN      A      145.239.83.154

;; Query time: 5 msec
;; SERVER: 192.168.68.2#53(192.168.68.2)
;; WHEN: Sun Jan 25 16:31:52 Central European Standard Time 2026
;; MSG SIZE rcvd: 61

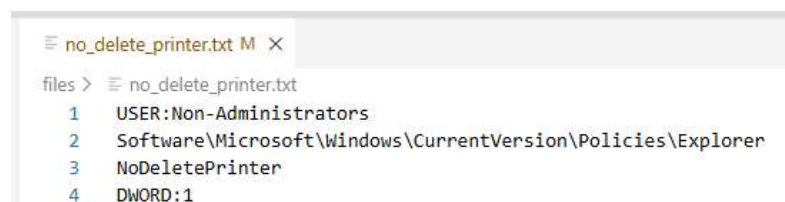
PS C:\Users\Administrator> |
```



Zrzut ekranu 31 Weryfikacja poprawności instalacji oprogramowania użytkowego na serwerze.

## Konfiguracja zasad grup lokalnych.

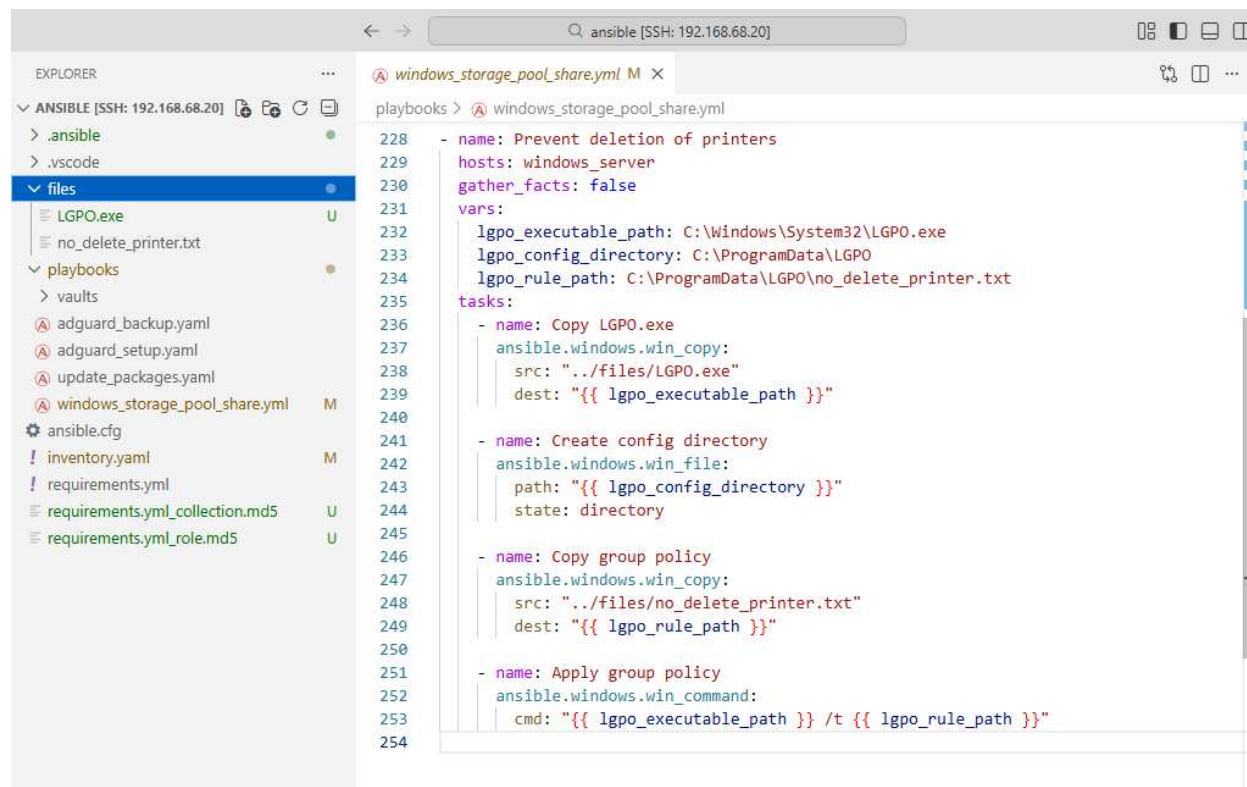
Na koniec skonfigurujemy jeszcze zasady grup lokalnych, aby ograniczyć użytkownikom nieadministracyjnym możliwość usuwania drukarek z systemu. Takiej konfiguracji moglibyśmy dokonać bezpośrednio z wykorzystaniem narzędzia *Microsoft Management Console*, jednak nie działa ono w trybie tekstowym. Z tego powodu, wykorzystam narzędzie *Local Group Policy Object Utility* dostępne do pobrania ze strony Microsoft pod linkiem <https://www.microsoft.com/en-us/download/details.aspx?id=55319>. Aby za jego pomocą skonfigurować odpowiednie zasady, należy przygotować plik w formacie *LGPO text*.



```
no_delete_printer.txt M X
files > no_delete_printer.txt
1 USER:Non-Administrators
2 Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
3 NoDeletePrinter
4 DWORD:1
```

Zrzut ekranu 32 Plik konfigurujący blokadę możliwości usuwania drukarek przez użytkowników nieadministracyjnych w formacie *LGPO text*.

Tak przygotowany plik, wraz z narzędziem *LGPO.exe* umieszczamy na komputerze pełniącym rolę węzła kontrolnego *Ansible*.



```
ansible [SSH: 192.168.68.20]
EXPLORER
  .ansible
  .vscode
  files
    LGPO.exe
    no_delete_printer.txt
  playbooks
    vaults
    adguard_backup.yaml
    adguard_setup.yaml
    update_packages.yaml
    windows_storage_pool_share.yaml M
  ansible.cfg
  ! inventory.yaml
  ! requirements.yaml
  requirements.yaml_collection.md5 U
  requirements.yaml_role.md5 U

playbooks > windows_storage_pool_share.yaml
228 - name: Prevent deletion of printers
229   hosts: windows_server
230   gather_facts: false
231   vars:
232     lgpo_executable_path: C:\Windows\System32\LGPO.exe
233     lgpo_config_directory: C:\ProgramData\LGPO
234     lgpo_rule_path: C:\ProgramData\LGPO\no_delete_printer.txt
235   tasks:
236     - name: Copy LGPO.exe
237       ansible.windows.win_copy:
238         src: "../files/LGPO.exe"
239         dest: "{{ lgpo_executable_path }}"
240
241     - name: Create config directory
242       ansible.windows.win_file:
243         path: "{{ lgpo_config_directory }}"
244         state: directory
245
246     - name: Copy group policy
247       ansible.windows.win_copy:
248         src: "../files/no_delete_printer.txt"
249         dest: "{{ lgpo_rule_path }}"
250
251     - name: Apply group policy
252       ansible.windows.win_command:
253         cmd: "{{ lgpo_executable_path }} /t {{ lgpo_rule_path }}"
254
```

Zrzut ekranu 33 Narzędzie *LGPO.exe* oraz plik konfiguracyjny zapisany na węźle kontrolnym *Ansible*. Skrypt konfigurujący zasadę grup lokalnych.

Możemy teraz przestać obydwie pliki na komputer docelowy, wykorzystując do tego celu moduł `ansible.windows.win_copy`, tworząc wcześniej ewentualne katalogi przy pomocy skryptu `ansible.windows.win_file`. Aby wdrożyć utworzoną zasadę grup lokalnych, możemy teraz uruchomić przestany przed momentem program z argumentem `/t <ścieżka_do_zasady>`.

```
● user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml

PLAY [Prevent deletion of printers] *****

TASK [Copy LGPO.exe] *****
changed: [windows_server]

TASK [Create config directory] *****
changed: [windows_server]

TASK [Copy group policy] *****
changed: [windows_server]

TASK [Apply group policy] *****
changed: [windows_server]

PLAY RECAP *****
windows_server      : ok=4   changed=4   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

● user@ansiblebieniek:~/ansible$ ansible-playbook ./playbooks/windows_storage_pool_share.yml

PLAY [Prevent deletion of printers] *****

TASK [Copy LGPO.exe] *****
ok: [windows_server]

TASK [Create config directory] *****
ok: [windows_server]

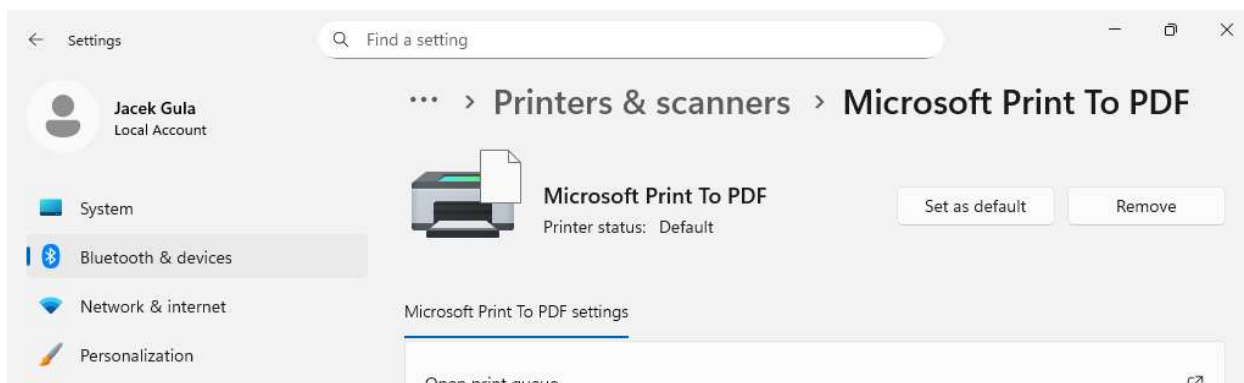
TASK [Copy group policy] *****
ok: [windows_server]

TASK [Apply group policy] *****
changed: [windows_server]

PLAY RECAP *****
windows_server      : ok=4   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

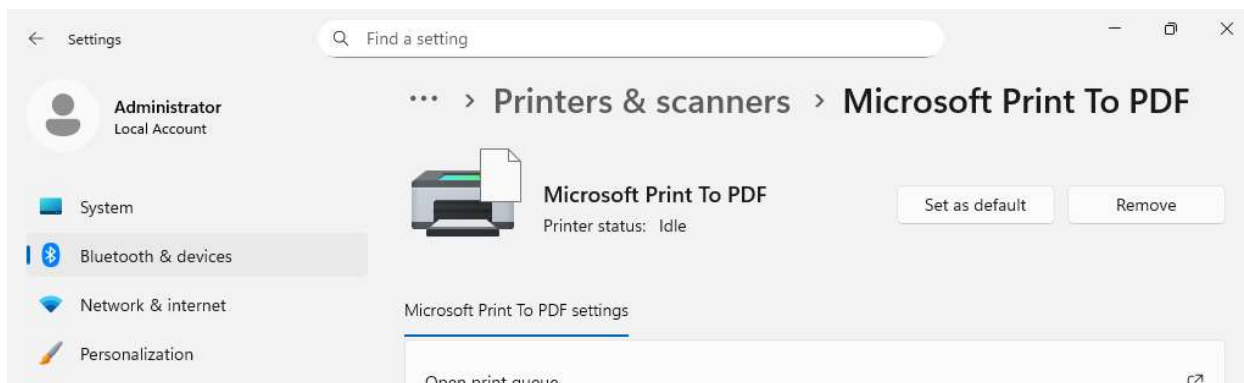
○ user@ansiblebieniek:~/ansible$
```

Zrzut ekranu 34 Dwukrotne wykonanie playbooka konfigurującego zasady grup lokalnych.



Zrzut ekranu 35 Brak opcji usuwania drukarek po wprowadzeniu zasady grup lokalnych.

Oczywiście użytkownicy administracyjni dalej posiadają możliwość usuwania drukarek z systemu.

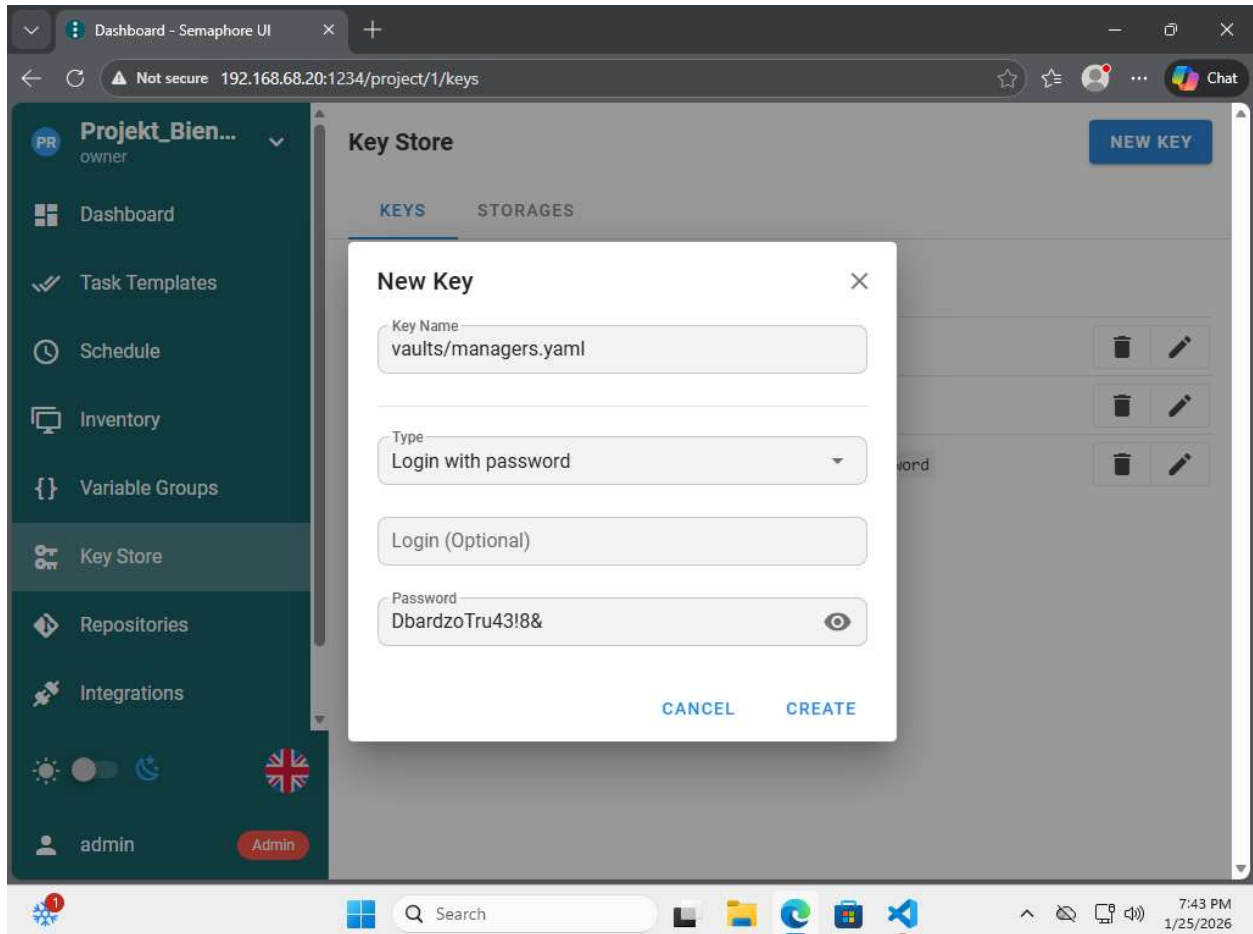


Zrzut ekranu 36 Opcja usuwania drukarek widoczna dla użytkowników administracyjnych.

## Konfiguracja Semaphore UI.

Po przygotowaniu w pełni działającego *playbooka* możemy dodać konfigurację do *Semaphore UI*, aby móc ją w przyszłości w łatwiejszy sposób uruchamiać.

Przed utworzeniem zadania dodajmy skarbiec z danymi kierowników.



Zrzut ekranu 37 Dodanie skarbcza wraz z hasłem do Key Store w Semaphore UI.

Następnie dodajmy nowy szablon zadania, wskazując utworzony przed momentem skarbiec.

**New template 'Ansible Playbook'**

**Common options**

- TASK** BUILD DEPLOY
- Name \*  
Configure storage pool and network s
- Path to playbook file \*  
/ansible/playbooks/windows\_storage
- Inventory \*  
Inventory
- Repository \*  
Local
- Variable Group \*  
Main
- View

**Advanced options**

Survey Variables

+ Add variable

Vault ID (optional)

Type  
Password

Vault Password  
vaults/managers.yaml

CANCEL ADD

**Ansible options**

Limit

+ Add limit

Tags

+ Add tag

Skip tags

+ Add skipped tag

Vaults

+ Add Vault

**Ansible prompts**

☐ Limit ☐ Tags ☐ Skip tags

☐ Debug

CANCEL CREATE

Zrzut ekranu 38 Utworzenie nowego szablonu zadania na podstawie utworzonego wcześniej playbooka.

Możemy nareszcie uruchomić utworzony skrypt przyciskiem „Run”.

**Projekt\_Bieniek** owner

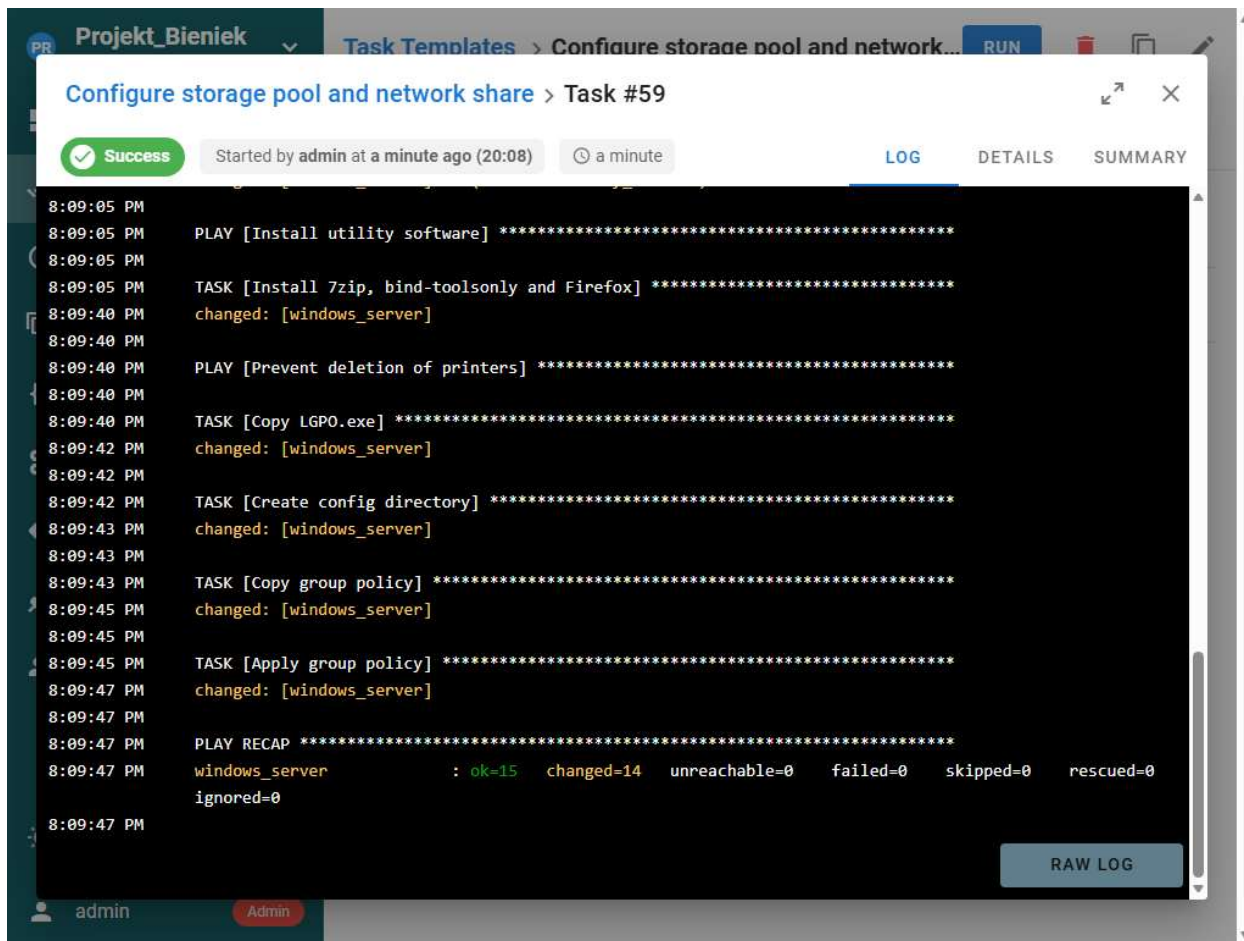
**Task Templates** > Configure storage pool and network... **RUN**

Empty

**TASKS** DETAILS

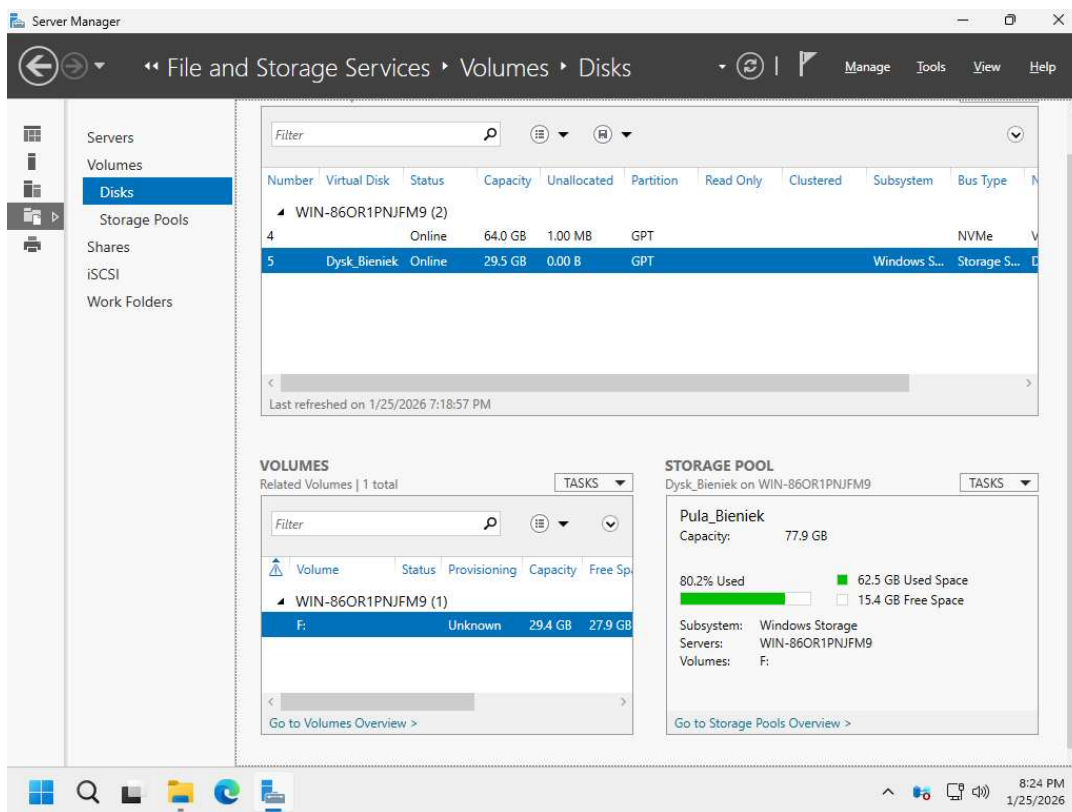
TASK ID	VERSION	STATUS	USER	START	DURATION
No data available					

Zrzut ekranu 39 Uruchomienie zadania z panelu Semaphore UI.

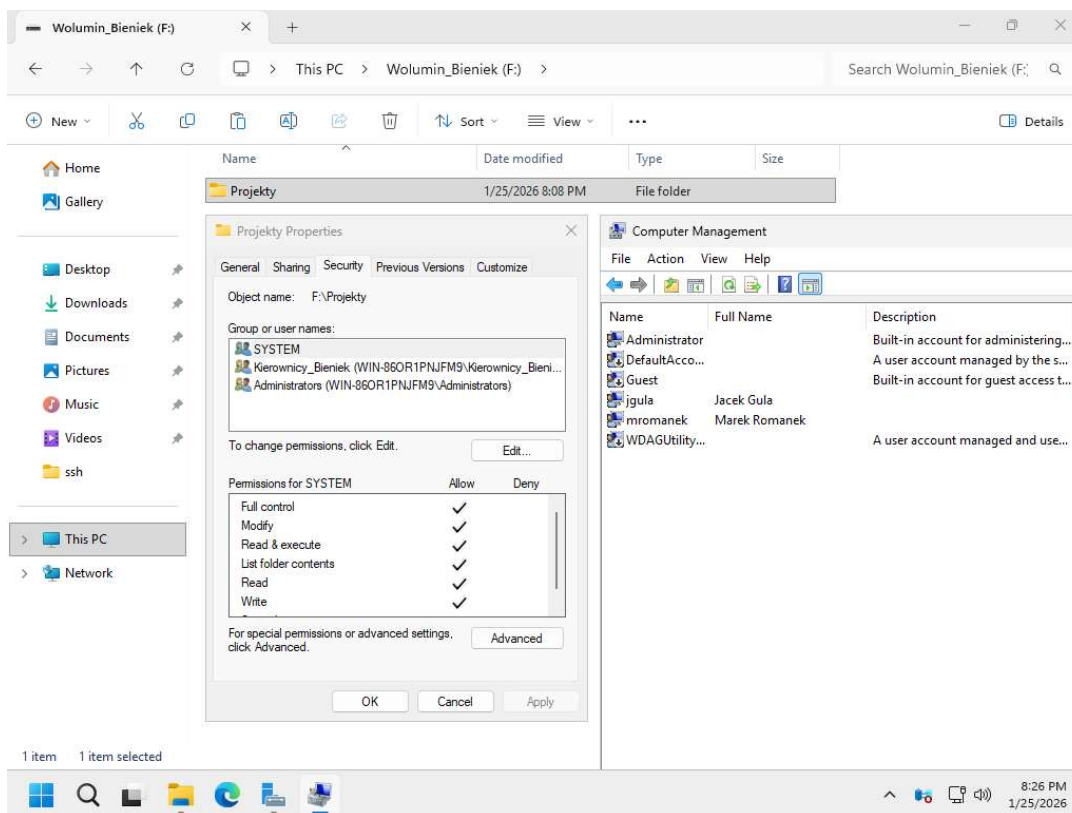


Zrzut ekranu 40 Zakończony sukcesem uruchomienie.

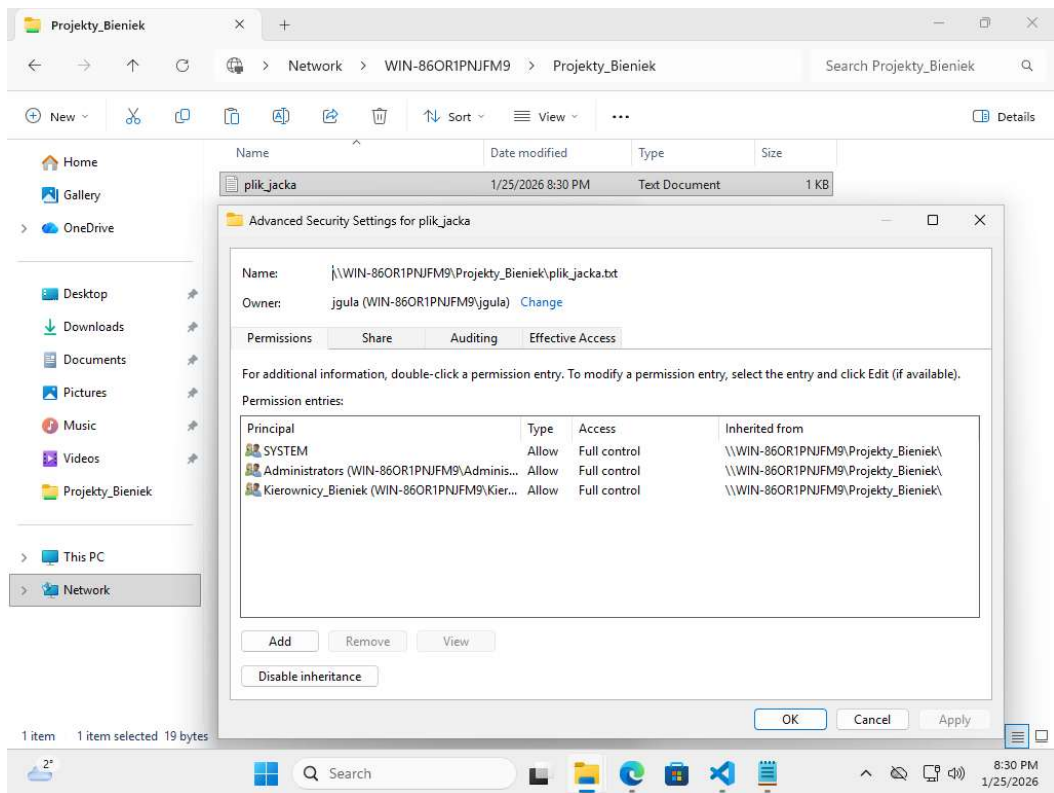
Jak widać wszystko zadziałało zgodnie z planem, a komputer z systemem Windows Server został poprawnie skonfigurowany.



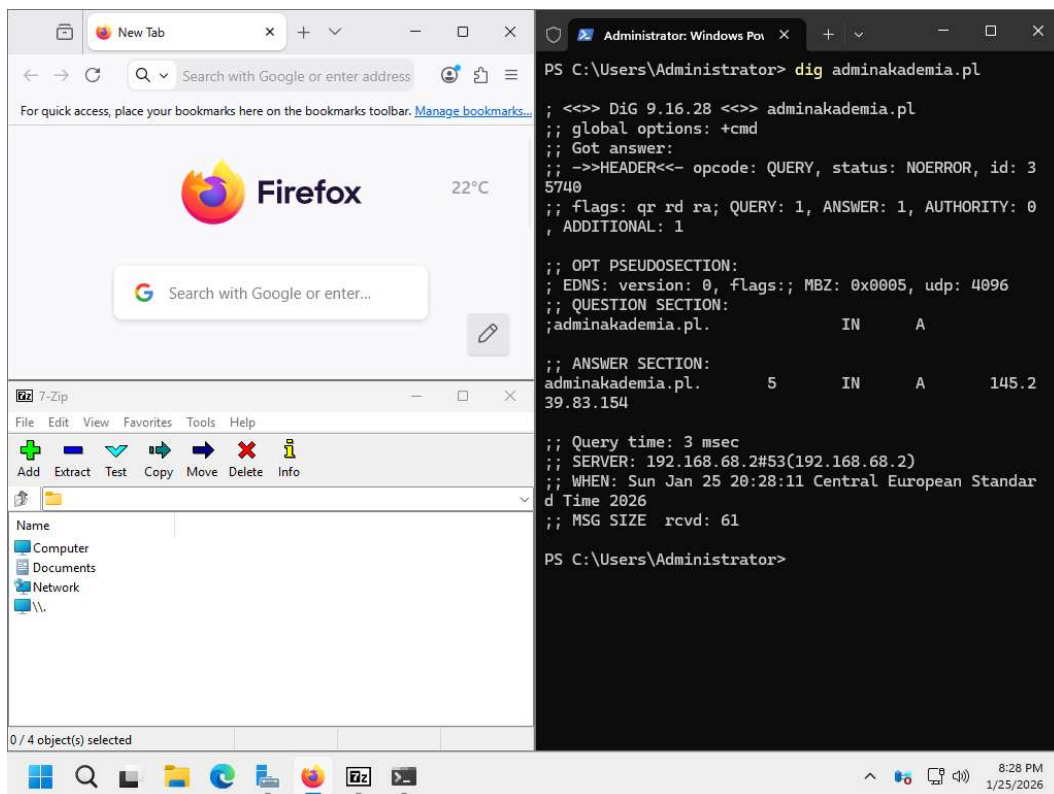
Zrzut ekranu 41 Utworzona pula dyskowa, dysk wirtualny oraz wolumin.



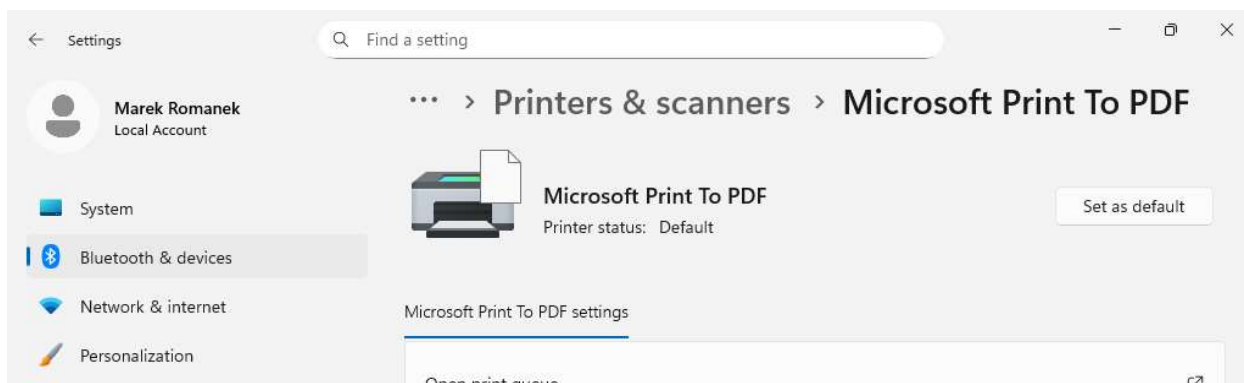
Zrzut ekranu 42 Udostępniony w sieci folder oraz skonfigurowana lista ACL.



Zrzut ekranu 43 Uzyskany z komputera klienckiego dostęp do udziału sieciowego.



Zrzut ekranu 44 Zainstalowane oprogramowanie użytkowe.



Zrzut ekranu 45 Zablockowana opcja usuwania drukarek z konta nieadministracyjnego.