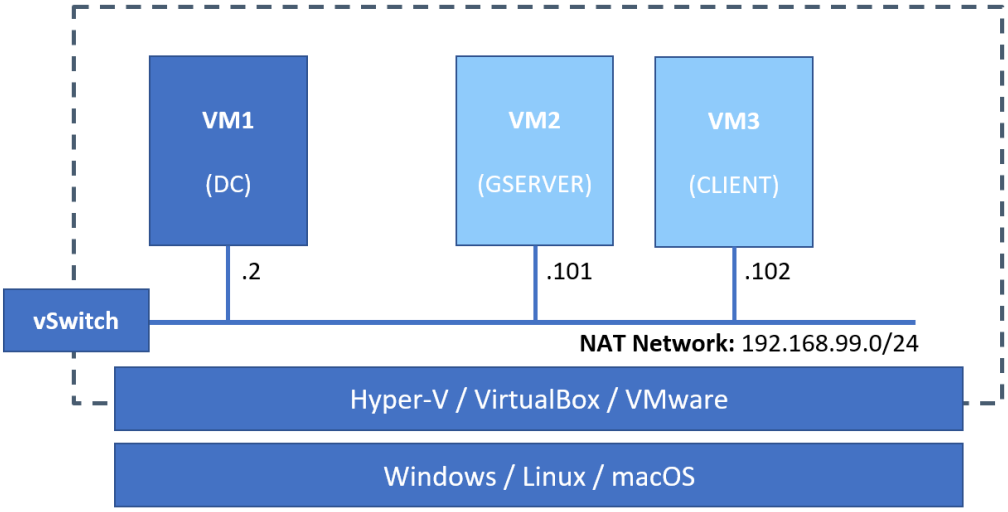
Practice M6: System Monitoring and Maintenance

We will use just part of the lab environment from the last module. Effectively we can do it on a standalone computer, no matter is it part of a domain or not.

We can safely assume that the following tasks are being executed on the Domain Controller (DC) from the last module:



Part 1: Troubleshooting and Monitoring

GUI Tools

Let's first explore the Task Manager and examine our system:

- The easiest way to open **Task Manager** is to press **Ctrl + Shift + Esc**. Alternative combination is to press **Ctrl + Alt + Del** (or if on a VM, depending on the mode and software, **Del** can be substituted with **End** or **Ins**) and from the set of options, choose **Task Manager**. Third option is to right-click on the taskbar and select **Task Manager** from the menu
- Depending on the system (version, is it core or with desktop experience, and is it server or client) the initial look and feel of the **Task Manager** can vary. For example, if we call it on a **Windows 10**, it will open in a "slim" mode, and in order to see everything, we must click on **More details**
- Go on, click on every tab, examine the information there, and adjust the settings where applicable in order to have another view on the subject

Now, open **Performance Monitor**

- Open it either from:
  - **Server Manager** – **Tools** – **Performance Monitor**
  - **Start Menu** – **Windows Administrative Tools** – **Performance Monitor**
  - Press **Alt + R** and enter **perfmon**
- Now that we are in, let's explore the interface
- The initial mode that opens is the **Overview**. Here we can see information about **CPU**, **Memory**, **Disk**, and **Network**
- More interesting would be to select **Performance Monitor** in the left section
- Initially we see one of the most popular performance counters - **% Processor Time**
- Let's add another counter, by clicking on the **green plus** button, or by pressing **Ctrl + N**

- For example, we can add **% Idle Time**, or something else. First select the counter, then click the **Add** button, and then **OK**
- Next stop is the **Data Collector Sets**. We can create our own or examine of the existing
- Select the **Server Manager Performance Monitor** set in the **User Defined** folder
- If you want to examine what it contains, you can double click on the **Performance Counters** item
- Beside the included counters, you can see for how long this set will be run, and where the result will be stored
- Close the dialog box
- Right click on the collector set and select **Start** and you will see an error message. In order to start this collector set, you must open the **Server Manager**, select the server in question, scroll until you reach the **Performance** section, there select the server and from the context menu choose **Start**. Please keep in mind that this will run for 24 hours if not stopped
- While it is collecting its data, let's create our own:
    - Select the **User Defined folder**
    - From the context menu choose **New > Data Collector Set**
    - For name enter **Demo**
    - Then select **Create manually (Advanced)**
    - Click **Next**
    - Leave the **Create data logs** selected
    - Select **Performance counter**
    - Click **Next**
    - Click **Add**
    - By default, the **Processor** counter is selected
    - Click on the down-arrow to expand the properties
    - Select **% Processor Time**
    - Ensure that in the instances section the **_Total** option is selected
    - Click **Add>>**
    - Click **OK**
    - Leave everything as it is and click **Next**
    - Then click **Next**, and finally **Finish**
    - Now select our new collector set and click **Start**
    - Wait for 10 seconds and then stop it
    - Go to the folder **Reports > User Defined > Demo** and double click on the report to see what was collected

There is one more hidden functionality in the **Performance Monitor** tool – it is called **Resource Monitor Report** and it can be reached by either:

- Executing **perfmon /report** on the command line
- With **Performance Monitor** open, go to **Data Collector Sets > System Diagnostics**, click **Start**, and then go to **Reports > System > System Diagnostics**
- Once open, examine its contents

One more report exists, this time it is for system reliability, and again it can be seen by either:

- Executing **perfmon /rel** on the command line
- Or in the **Control Panel** go to **System and Security > Security and Maintenance > Reliability Monitor**
- Once open, check the reliability of your system

Another useful tool is the **Resource Monitor**:

- It can be launched by any of the following ways:
    - By clicking **Open Resource Monitor** in the **Performance** tab of the **Task Manager**
    - By clicking **Resource Monitor** in the **Tools** menu of the **Server Manager**
    - By executing either **resmon** or **perfmon /res** on the command line
- Let's examine the tabs and their sections

Let's open **Event Viewer** and examine the captured events:

- Typically, there are many ways to open the tool, for example using the **Server Manager**, command line (**eventvwr**), **Start menu**, and etc.
- Once in, we can see that on the left part of the window, there are several nodes
- Let's open the **Windows** node and examine the events in the **Applications** leaf

Check the following places for interesting event codes and their meaning:

- AD related events to monitor
  https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor
- Security related events
  https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/
- Interesting Windows Event IDs – Malware/General Investigation
  https://support.sophos.com/support/s/article/KB-000038860?language=en_US
- Netsurion Event Tracker
  https://kb.eventtracker.com/

Use one or more of the above to check what events with ID **6005**, **6006**, **6008**, and **1074** means

CMD Tools

Let's try several command-line utilities. Open a CMD shell and:

- Type **systeminfo** and examine the provided information
- Now type **tasklist** and compare the result with the information in **Task Manager**
- Let's add one parameter, now execute **tasklist /v**
- And now, let's filter the information by user - **tasklist /FI "USERNAME eq NT AUTHORITY\SYSTEM" /v**
- Now open a browser window and execute – **tasklist /FI "IMAGENAME eq iexplore.exe"**
- If by chance we want to close or kill certain process, there is an easy way to do it. The tool for the job is **taskkill**
- For example, if we want to kill the two iexplore.exe process from the previous step, we can do it with: **taskkill /IM iexplore.exe**

We can work with performance counters on the command line as well:

- The utility for the task is **typeperf**
- Let's see what parameters are expected by typing **typeperf /?**
- Now that we know how to see all performance counters, let's ask for the list with **typeperf /q**
- The list is long, so we can pause it one screen at a time by **typerperf /q | more**
- Even this way is not quite usable, so we can filter and pause at the same time with: **typeperf -q | find /i "Processor(*)" | more**
- And finally, let's ask for a counter: **typeperf "\Processor(*)\% Processor Time"**
- Press **Crtl + C** to stop the monitoring of the counter
- We can add few more parameters if we want to achieve the following – get 5 measures with 3 seconds delay:

typeperf "\Processor(*)\% Processor Time" -si 5 -sc 3

PowerShell

As usual most of the functionalities have their equivalent in PowerShell as well. Let's see few examples about the counters:

- Get set of default counters on the local computer with:

Get-Counter

- To get a specific counter on the local computer execute:

Get-Counter '\processor(_total)\% processor time'

- We can ask for a number of observations every X seconds, for example 5 observations, every 5 seconds:

Get-Counter '\processor(_total)\% processor time' -SampleInterval 5 -MaxSamples 5

And now few more, but this time about events:

: https://docs.microsoft.com/en-us/sysinternals/

- We can ask for all **Application** events with:

Get-EventLog Application

- Or filter them by type:

Get-EventLog Application -EntryType Error

- By utilizing the piping mechanism, we can filter the results even further:

Get-EventLog Application -EntryType Error | Where -Property Source -Like 'Application*'

Windows Sysinternals Tools

We can extend our administrator toolset by downloading few or all the freely available **Sysinternals** utilities. To learn more about then, go to: https://docs.microsoft.com/en-us/sysinternals/

Or you can download the whole pack directly from https://download.sysinternals.com/files/SysinternalsSuite.zip

Part 2: Backup and Restore

Backup (GUI)

As a prerequisite for this section, we should add two more hard drives. The size can be as small as 10 or 20 GB with thin provisioning. Ensure that the first one is split in two equal parts, each formatted and mounted, and the other one is not.

Once that you have the drive, open the **Disk Management** tool and fulfil the prerequisites.

Now let's install the feature:

- Start **Server Manager** if not already started
- Choose **Add Roles and Features** from the **Manage** menu
- Then click **Next**
- Then do not make any changes and click **Next**
- Leave the default selection (current server) and click **Next**
- Do not modify the selection and click **Next**
- Then select **Windows Server Backup** and continue with **Next**
- Click **Install**

- And finally, click **Close**

Now that we have software installed, let's do see two usage scenarios. First would be to back up the system state:

- Start the **Windows Server Backup** tool either from the **Server Manager**, or from the **Start menu > Windows Accessories**
- Select **Local Backup** option on the left side of the window, and then click on **Backup Schedule** in the right section
- Read the notes and click **Next**
- Select **Custom** and click **Next**
- Click **Add Items**
- Select **System state** and click **OK**
- Click **Next**
- Prepare the schedule and click **Next**
- Click on **Show All Available Disks**
- Select the not mounted one and click **OK**
- Then, select the added disk, and click **Next**
- Read the warning message and click **Yes**
- Examine the backup job settings and click **Finish**
- After the disk is initialized and the job is created, click **Close**

Okay, we can wait for the time to come, or we can force a one-time backup based on the created job:

- With the **Windows Server Backup** tool opened, select **Local Backup** on the left
- Then in the right section choose **Backup Once**
- Leave the **Scheduled backup options** choice selected and click **Next**
- Then click **Backup**
- Watch the backup process or click **Close**

If do not have the space, or we do not want the **System state** backup to happen, we can stop it:

- You can return to the job by clicking twice on it in the **Messages** area in the main windows
- In the open dialog, click **Cancel**, because this backup will take long, and will require big amount of space
- Furthermore, you can delete the schedule by clicking on the **Backup Schedule** option in the right section
- Then select **Stop** backup and click **Next**
- Finally, click **Finish**
- And confirm with **Yes**
- As last step click **Close**

Do some preparation before next task. You can either create a folder **C:\Important** and put there some files, or execute the following set of commands:

```
ForEach ($D in ('BUDGET','CODE','PLAN','SALES','ZZZ')) {
  New-Item -Type Directory -Path "C:\Important\$D";
  New-Item -Type File -Path "C:\Important\$D\readme.txt"
}
```

Let's now create a backup of the **C:\Important** folder to a mounted drive:

- With the **Windows Server Backup** tool open, select **Local Backup** in the left section, and click on Ba**ckup Once** in the right

- Accept the default selection (**Different options**) and click **Next**
- Select **Custom** and click **Next**
- Click **Add Items**
- Open the node for drive **C:** and select **Important** folder, then click **OK**
- Click **Next**
- Accept the default selection (**Local drives**) and click **Next**
- From the drop-down list select the mounted volume you added earlier and click **Next**
- Click **Backup**
- Monitor the process or click **Close**

Now let's try to restore the folder:

- First let's delete the folder or few of its files – it's your choice
- With the **Windows Server Backup** tool open, select **Local Backup** in the left section, and click on **Recover** in the right
- Accept the default option (**This server (DC)**) and click **Next**
- Select a date with a backup and click **Next**
- Leave the selection (**Files and folders**) and click **Next**
- Mark the folder in the **Available items** section and click **Next**
- On the next screen adjust options if needed and click **Next**
- Click **Recover**
- Monitor the process or click **Close**

Backup (PowerShell)

As usual, we can do more or less the same on the command line. Let's experiment with **PowerShell**:

- Execute the following to create a new policy object:

```
$Policy = New-WBPolicy
```

- Then crate a backup source:

```
$Filespec = New-WBFileSpec -FileSpec "C:\Important"
```

- Associate the backup source and the policy:

```
Add-WBFileSpec -Policy $Policy -FileSpec $FileSpec
```

- Create a backup location:

```
$BackupLocation = New-WBBackupTarget -VolumePath "Y:"
```

- Associate the backup target and the policy:

```
Add-WBBackupTarget -Policy $Policy -Target $BackupLocation
```

- Set that the backup policy will use **Volume Shadow Copy Service (VSS)** copy backups:

```
Set-WBVssBackupOptions -Policy $Policy -VssCopyBackup
```

- Start the backup:

```
Start-WBBackup -Policy $Policy
```

- Now, we can switch to the **Windows Server Backup** tool and monitor the process as well

Part 3: Scheduling

Scheduling (GUI)

Quite often we need to execute repetitive tasks, and usually we do not want this to happen with our interaction, that why there are functionalities like scheduling:

- Start Task Scheduler either from the **Tools** menu in the **Server Manager** or from the **Windows Administrative Tools** in the **Start menu**. There is also an option to start it on the command line by executing **taskschd.msc**
- Get familiar with its interface
- Let's define simple and not very useful task – for example, to start a browser and navigate it to **SoftUni** web page
- Select the **Task Scheduler (Local)** in the left section
- Click on **Create Basic Task** option in the right section
- For name enter **Demo** for example, and then click **Next**
- Leave the **Daily** option selected and click **Next**
- Adjust the time and recurrence, and click **Next**
- Leave the **Start a program** option selected and click **Next**
- Click **Browse** and navigate to **"C:\Program Files (x86)\Internet Explorer\iexplore.exe"**
- Then in the **Add arguments** fill in www.softuni.bg
- Then click **Next**
- And then **Finish**
- Now, you should see our task in the active tasks list
- And eventually when the time comes, we will see the browser open and trying to load the requested page
- We can explore what scheduled tasks exist on our system by opening a command shell (**cmd.exe**) and executing a **schtasks.exe**

Scheduling (PowerShell)

Let's do a similar thing, but this time in PowerShell. First, we will explore a bit:

- To see all defined scheduled tasks, execute:

Get-ScheduledTask

- Then let's use a filter to narrow the list.

Get-ScheduledTask | Where State -Eq "Running"

- Examine properties of the task we created earlier, with:

Get-ScheduledTask -TaskName "Demo" -TaskPath "\" | Select *

Get-ScheduledTask -TaskName "Demo" -TaskPath "\" | Select -ExpandProperty Actions

Now, create a new scheduled task:

- First, define the action:

$A = New-ScheduledTaskAction -Execute "C:\Program Files (x86)\Internet Explorer\iexplore.exe" -Argument "http://dir.bg"

- Then create the trigger:

$T = New-ScheduledTaskTrigger -AtLogOn

- Associate the action and the trigger by creating a new task:

$D = New-ScheduledTask -Action $A -Trigger $T

- Now register the task:

Register-ScheduledTask T1 -InputObject $D

- Everything is ready. If we sign out and the sign in again, the task should be triggered

And now, let's create one more scheduled task. It will store a counter value in a text file:

- That tracks the following counter:

Get-Counter -Counter "\Memory\Available MBytes"

- Modify the command to this:

Get-Counter "\Memory\Available MBytes" | Foreach-Object {$_.Timestamp.ToString() + ' => ' + $_.CounterSamples.CookedValue[0]}

- And one final preparation – add the following at the end to store its value in a file:

Out-File -FilePath 'C:\Temp\Memory.log' -Append

- The final line should be:

Get-Counter "\Memory\Available MBytes" | Foreach-Object {$_.Timestamp.ToString() + ' => ' + $_.CounterSamples.CookedValue[0]} | Out-File -FilePath 'C:\Temp\Memory.log' -Append

- Save the command as a script under **C:\Scripts\RAM.ps1**

To create the actual schedule, we can either follow the graphical approach, or use PowerShell. The GUI way is:

- Now that we have all components, let's open the **Task Scheduler** and create a new task that will run **every 2 minutes:**
  - Choose **Create Task** in the right section
  - For name enter **RAM Monitor**
  - Switch to **Triggers** tab and click **New**
  - In **Settings** select **Daily**
  - Select **Repeat task every** and enter **2 minutes**
  - In the field **for a duration of** set **Indefinitely**
  - Click **OK**
  - Switch to **Actions** tab and click **New**
  - Navigate to **C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe**
  - In the **Add argument** section type the path to the script – **C:\Scripts\RAM.ps1**
  - Click **OK**
  - And then click **OK** again
- Examine the result and eventually stop the task

The PowerShell way is:

- Same can be done in PowerShell and will look like:
  - Define the action:
    ```
    $A=New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-NonInteractive -NoLogo -NoProfile -Windowstyle Hidden -File C:\Scripts\RAM.ps1"
    ```
  - Then a trigger:
    ```
    $T=New-ScheduledTaskTrigger -Once -At (Get-Date) -RepetitionInterval (New-TimeSpan -Minutes 2)
    ```

    - Finally, register the task:
      `Register-ScheduledTask -Action $A -Trigger $T -TaskName "RAM Monitor" -Description "Demo scheduled task"`
- We can delete the scheduled task by:
  `Unregister-ScheduledTask -TaskName "RAM Monitor"`

Finally, let's create a scheduled job, which will be exactly like the scheduled task above. Do not forget to launch the PowerShell session with elevated privileges. Enter consequently the following commands:

- Define the trigger:
  `$T=New-JobTrigger -Once -At (Get-Date) -RepetitionInterval (New-TimeSpan -Minutes 2) -RepeatIndefinitely`
- Then the options if any:
  `$O=New-ScheduledJobOption -DoNotAllowDemandStart -MultipleInstancePolicy IgnoreNew`
- Finally, register the job:
  `Register-ScheduledJob -Name "RAM Job" -FilePath "C:\Scripts\RAM.ps1" -Trigger $T -ScheduledJobOption $O`

Now, open **Tash Scheduler** and navigate to **\Microsoft\Windows\PowerShell\ScheduledJobs**. The job should appear there. The same can be achieved with **Get-ScheduledJob**

In a similar way, we can delete the scheduled job with:

`Unregister-ScheduledJob -Name "RAM Job"`