ABB

Ping Monitoring System

**Tables of Contents**

[1. Purpose and Basic Description 3](#_Toc108517489)

[1.1. Notes [Before going forward, please read all points below]: 4](#_Toc108517490)

[1.1.1. Unblocking Files/Folders 4](#_Toc108517491)

[1.1.2. Output File [ipoutput.txt] 7](#_Toc108517492)

[2. Python Script and PowerShell Script 9](#_Toc108517493)

[2.1. PowerShell Script 9](#_Toc108517494)

[2.1.1. PowerShell Editor Installation [Optional Download] 9](#_Toc108517495)

[2.1.2. Running the PowerShell Script 11](#_Toc108517496)

[2.2. Python Script 13](#_Toc108517497)

[2.2.1. Python Installation [Without Code Editor] 13](#_Toc108517498)

[2.2.2. Python and Code Editor Download 14](#_Toc108517499)

[2.2.3. Running the Python Script 16](#_Toc108517500)

[3. OneDrive Business 18](#_Toc108517501)

[3.1.1. Important Note 18](#_Toc108517502)

[4. Windows Task Scheduler 20](#_Toc108517503)

[4.1. PowerShell Script 20](#_Toc108517504)

[4.2. Python Script 23](#_Toc108517505)

[4.3. OneDrive Business 27](#_Toc108517506)

[5. Microsoft Power Automate 30](#_Toc108517507)

[5.1. Microsoft Power Automate Flow 31](#_Toc108517508)

# Purpose and Basic Description

For our demonstration environment in the ABB Burlington offices, we wanted to establish a mechanism to get proactive notifications related to overall System / Infrastructure availability. The system is made up of:

* ESXi Host – running vmWare 7.0
  + vSphere VM running on the ESXi Host
  + Multitude of Windows based VM’s
  + Several Ubuntu/CentOS based VM’s

Since availability is the key aspect, we are looking to understand, and we are not concerned about performance monitoring, it was decided to architect a solution that was no-code, no-cost and could take advantage of existing ABB subscriptions and standard applications.

This documentation contains information about the Ping Monitoring System. The monitoring system monitors the infrastructure as described above by using a basic ping command. The solution is comprised of 4 parts:

* PowerShell Script or Python Script – Diagnostic Generation
* Windows Task Scheduler – Scheduling of Diagnostic Script
* Microsoft OneDrive Business – Transportation of Diagnostic output
* Microsoft Power Automate – Post Processing of Diagnostic output with workflow elements

Ping Monitoring System Flow Diagram

Diagram

Description automatically generated

How the Ping Monitoring System Works

1. The Task Scheduler will run the PowerShell/Python Script
2. The Task Scheduler will run OneDrive Sync task
3. The Ping Monitoring System files will now be updated in the OneDrive business folder
4. Microsoft Power Automate will pull the output file that was generated by the PowerShell Script/Python Script and apply to the configured flow to send appropriate email to ABB Employees.

**The project repository can be found at this** [**GitLab Link**](https://codebits.abb.com/abb-ping-monitoring-system/abb-ping-monitoring-system)

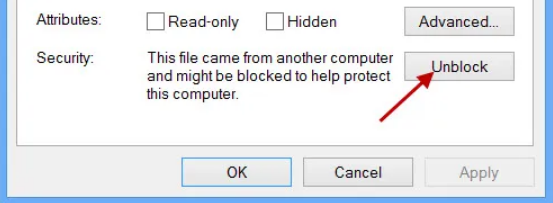
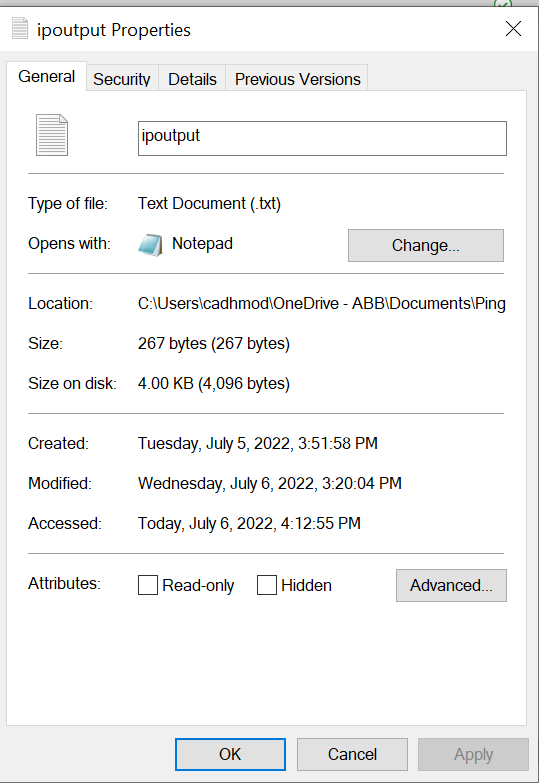
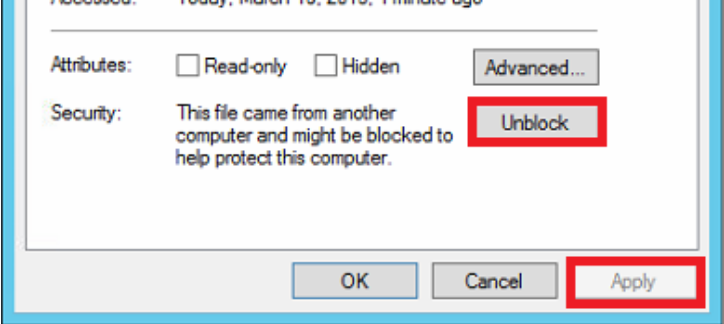
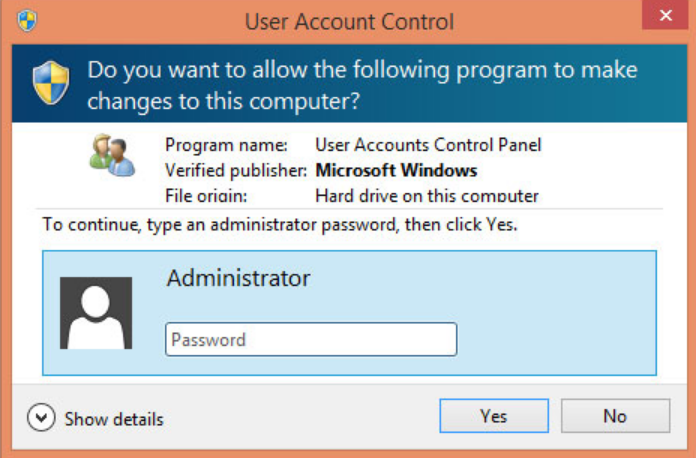
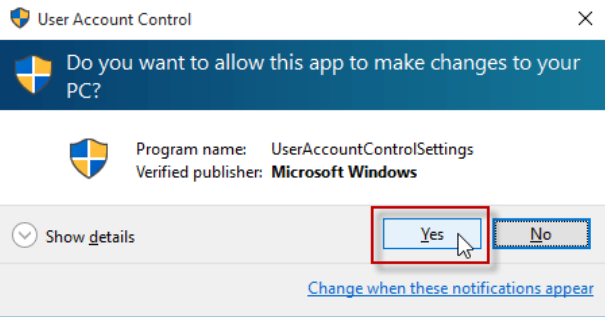
## Notes [Before going forward, please read all points below]:

* All the components are hosted on the **Digital Client Virtual Machine (VM)**
* For Windows Operating System [OS], either script will work:
  + PowerShell Script – No Installation/Less Installation Required
  + Python Script – Installations of Python is required
* This document covers both PowerShell Script and Python Script but is focused more on the PowerShell Script.

### Unblocking Files/Folders

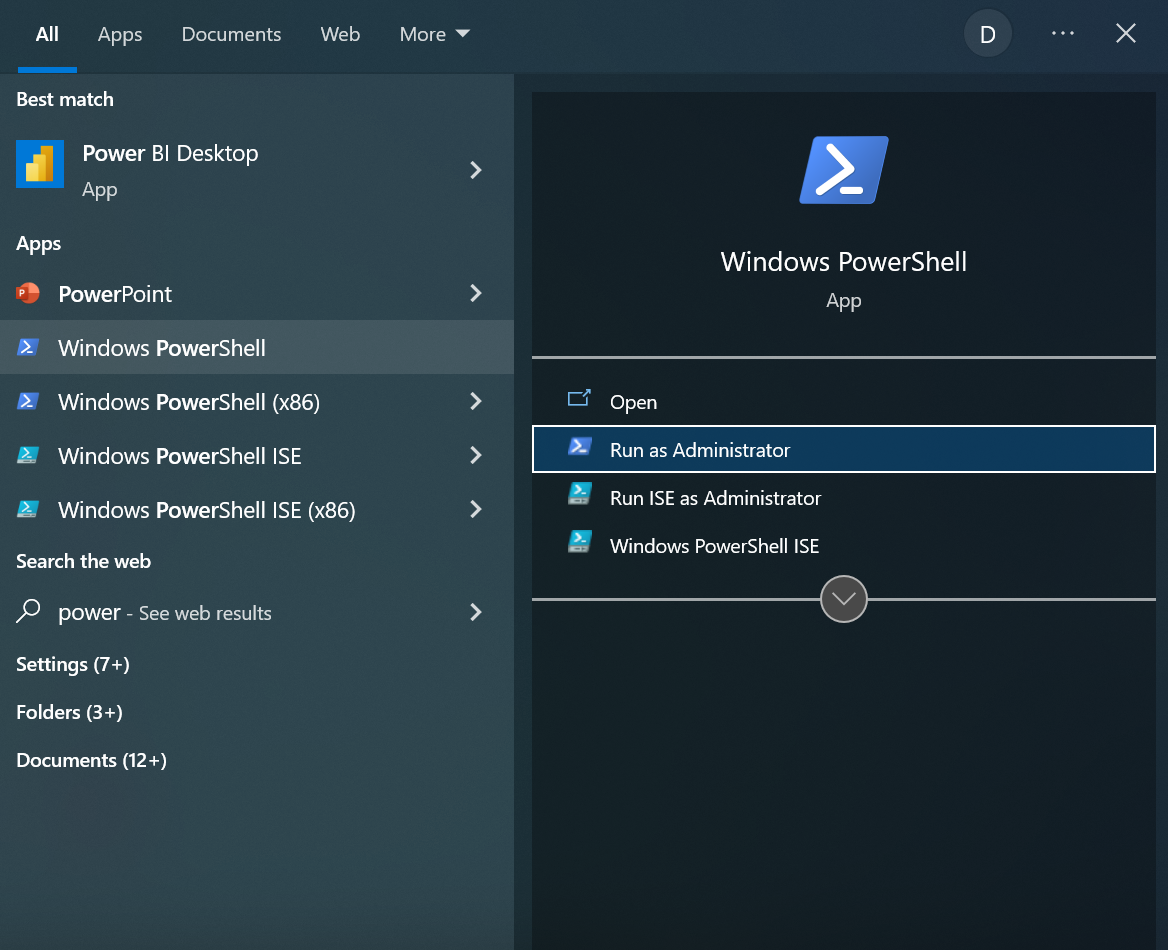
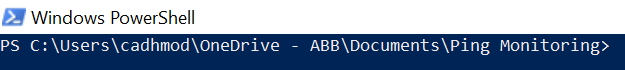
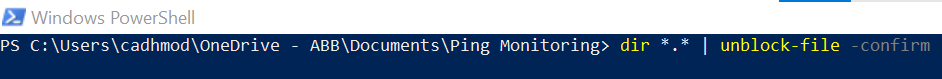
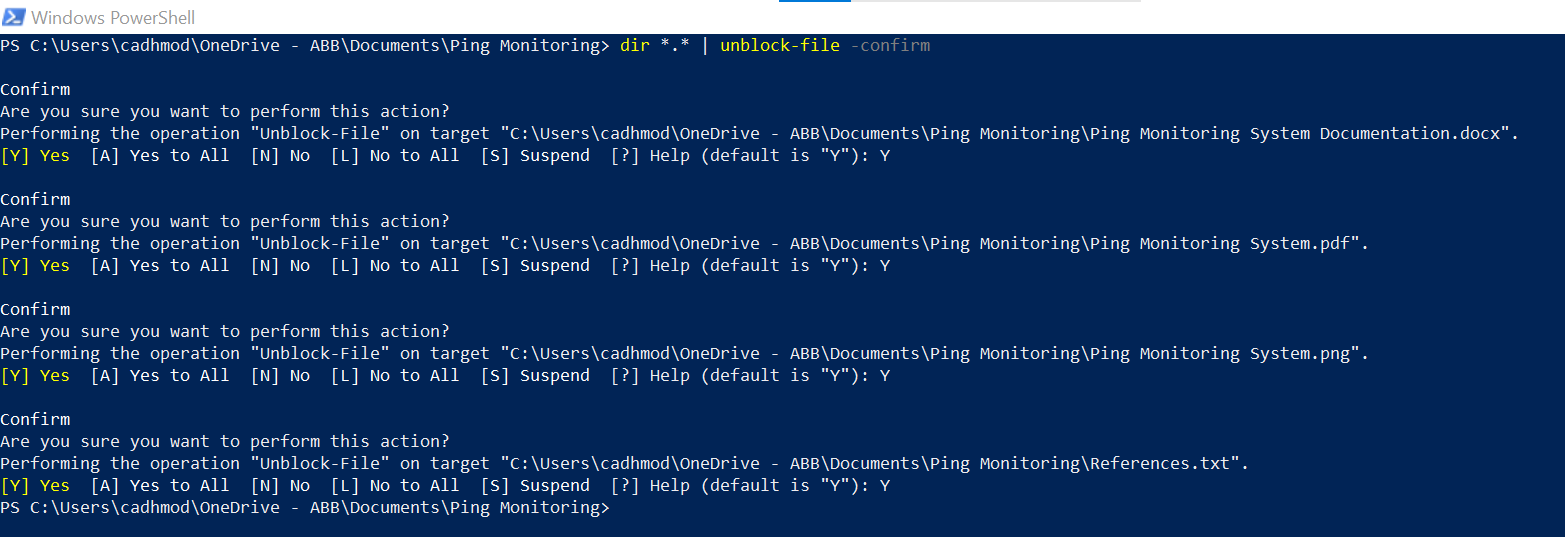
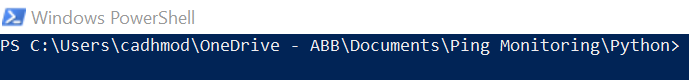
If the “Ping Monitoring” folder [which contains subfolders, scripts, text files, diagrams, this documentation] is downloaded, there is possibilities that files and folders are Blocked. Before going forward in the documentation, please follow the steps below to check Unblock/Block status for any files and folder and Unblock any files/folders if they are Blocked.

#### Unblocking Single File

* Step 1: Right Click on the File/Folder and click “Properties”
* Step 2: Click on the General tab
* Step 3: At bottom of the General tab, there will be Block/Unblock status of the file
  + If there is Block status, then there will be an option to switch to Unblock status 
  + If it’s already Unblocked than there will be no option to switch and there will also be no information about Block/Unblock 
* Step 4: If there is information to Unblock, click on the Unblock option, next click Apply, and then click okay 
* Step 5: A pop-up window regarding “Administrator Permission” could appear, please click continue [Enter the password if required for Administrator control]. For example,

#### Unblocking Multiple Files

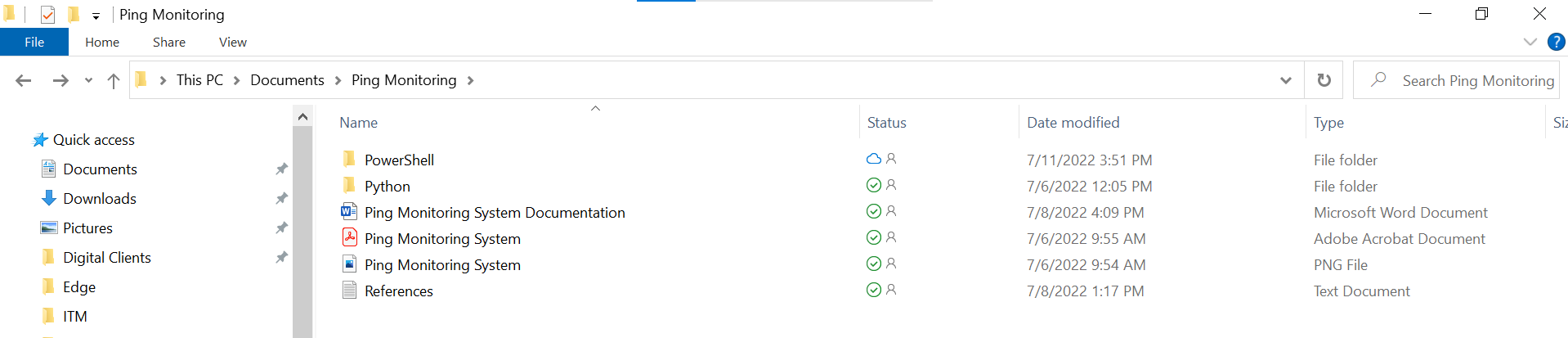
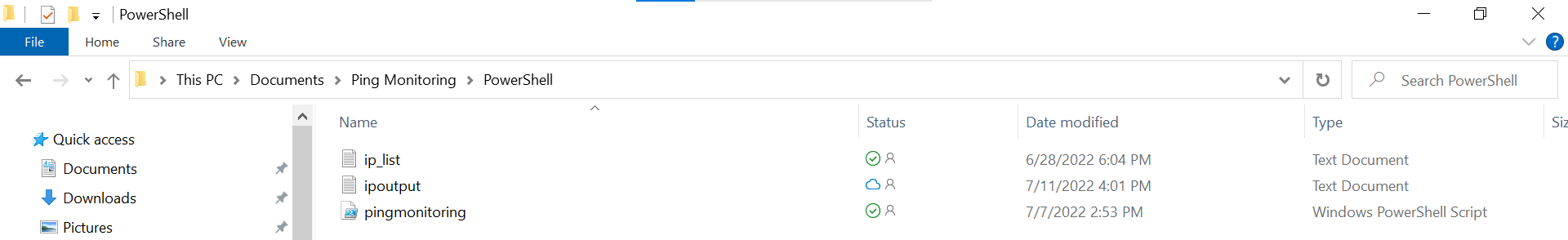
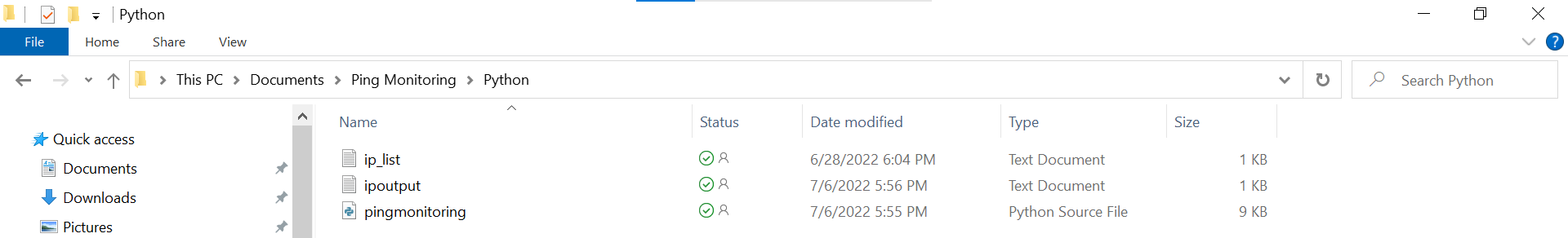
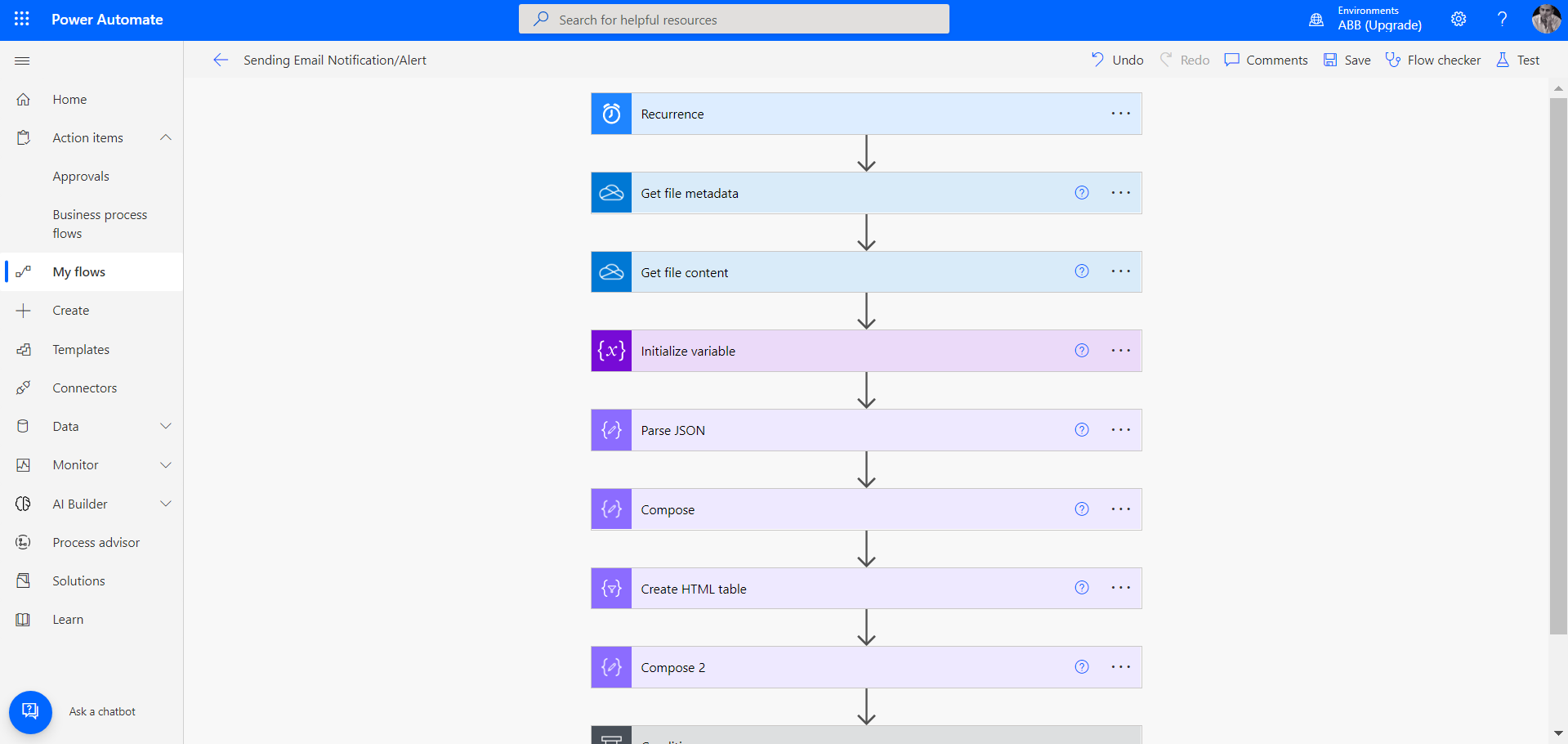
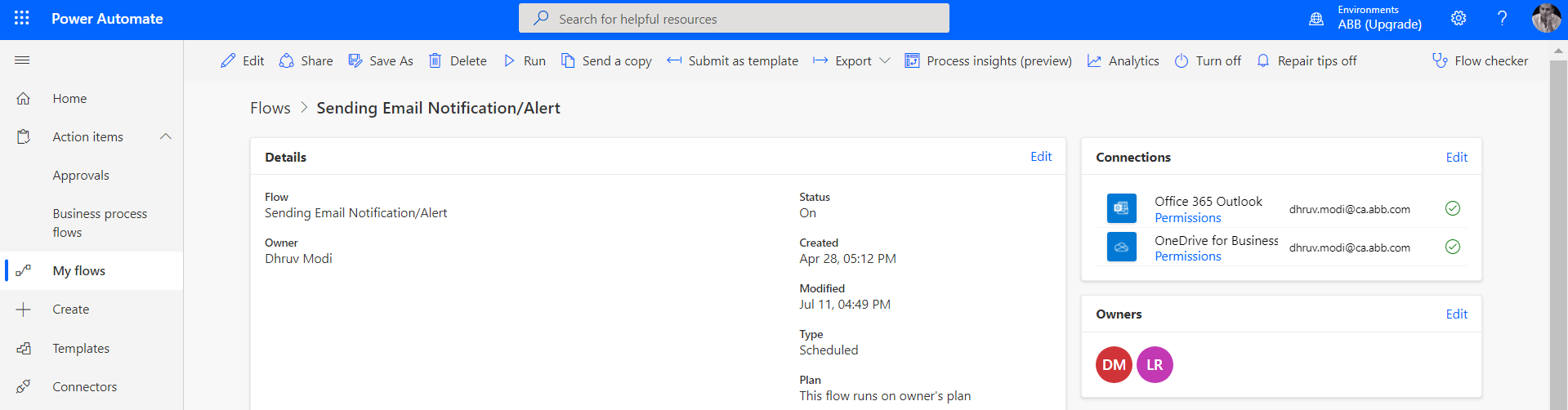
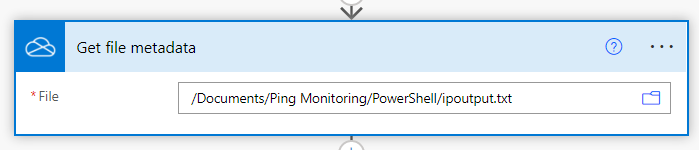
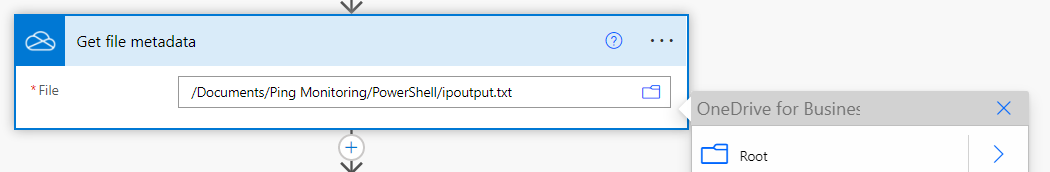
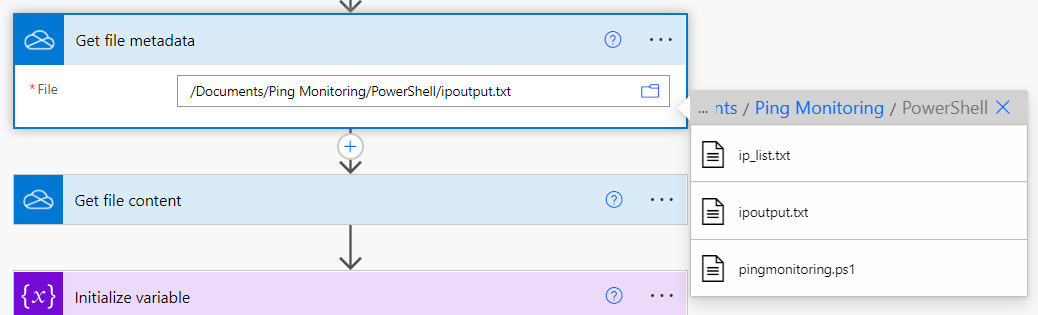
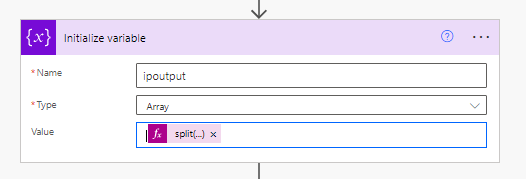
To Unblock multiple files, please follow these steps:

* Step 1: Open PowerShell as an Administrator, click on “Run as Administrator” 
* Step 2: Change the directory to the PowerShell Script with “cd” command [Add double quotes if spaces in the path, for example, name of folder 
* Step 3: Enter the command “dir \*.\* | unblock-file -confirm”, this give you the option to Unblock all files in this given path which is “Ping Monitoring” 
* Step 4: When a message arrives to confirm the Unblocking, Enter “Yes” or “Y for Unblocking each file in that folder 
* Step 5: Repeat step 3 and Step 4 for Python Subfolder and PowerShell subfolder by changing directories to those folders, for example “cd Python” and then repeat Step 3 and Step 4
* Step 6: Check if Unblocking is successful by following Step 1 – Step 3 from **Unblocking Single File [1.1.1.1.]**

### Output File [ipoutput.txt]

For the Ping Monitoring System to execute properly it is very important that the Output File, “ipoutput.txt” is part of the correct folders of the Ping Monitoring System, named correctly, and is attached to the Power Automate suitably.

**Note:** Please do not modify or change the Output file [ipoutput.txt] that is initially part of the Ping Monitoring System folder/Repository. This is the best approach to avoid the Output File [ipoutput.txt] errors listed below.

1. The Output File, “ipoutput.txt”, should be part of two folders, the PowerShell Folder and/or Python Folder
   1. 
   2. 
   3. 
2. The Output File correct naming: **ipoutput.txt [Text Format], please don’t change this name and its structure**
3. Please don’t type manually in the Output File [ipoutput.txt], the Output File [ipoutput.txt] syntax/structure is built specifically for our Power Automate
4. The Output File [ipoutput.txt] is connected to Power Automate through a special id, if the Output File [ipoutput.txt] is deleted from the folders, the connection will be broken and due to this reason, the Power Automate will not work which will lead to emails not being sent. If the file is deleted, we will have to link it back to the Power Automate, to do that follow these steps:
   1. Log in to Microsoft Power Automate and click on the “Edit” option at the top left corner for our flow
   2. Click once on the step “Get file metdata”, this will expand the step 
   3. Click on the folder icon on the right 
   4. Find the Output File [ipoutput.txt] from the right menu beside the step 
   5. Select the “ipoutput.txt”
5. Please don’t keep the Output File [ipoutput.txt] empty because the properties [Type and Value] of the “Initialize Variable” [Step 3] requires the Output file to have data. If the Output File [ipoutput.txt] is empty, the Step 3 [Initialize Variable] will not work. These will halt the Power Automate flow and the Power Automate won’t be able to send emails. If the Output File [ipoutput.txt] is empty, we can solve it by manually running the PowerShell/Python Script [**Please look at 2.1.2. for running PowerShell Script manually and 2.2.3. for Running Python Script manually**]. This will the fill the Output File [ipoutput.txt] with data

# Python Script and PowerShell Script

## PowerShell Script

The PowerShell Script section has three parts:

1. PowerShell Script (pingmonitoring.ps1)

2. Input Text File (ip\_list.txt)

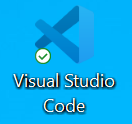
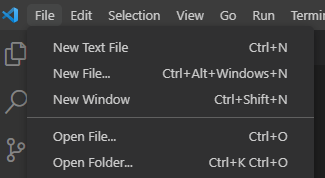
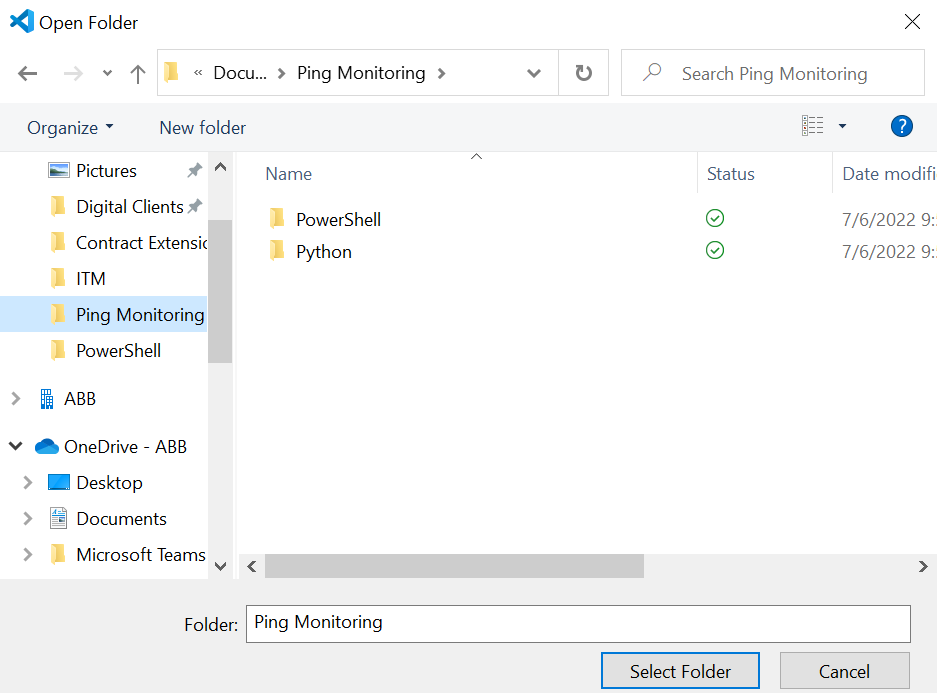
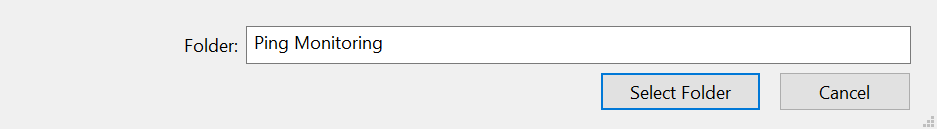
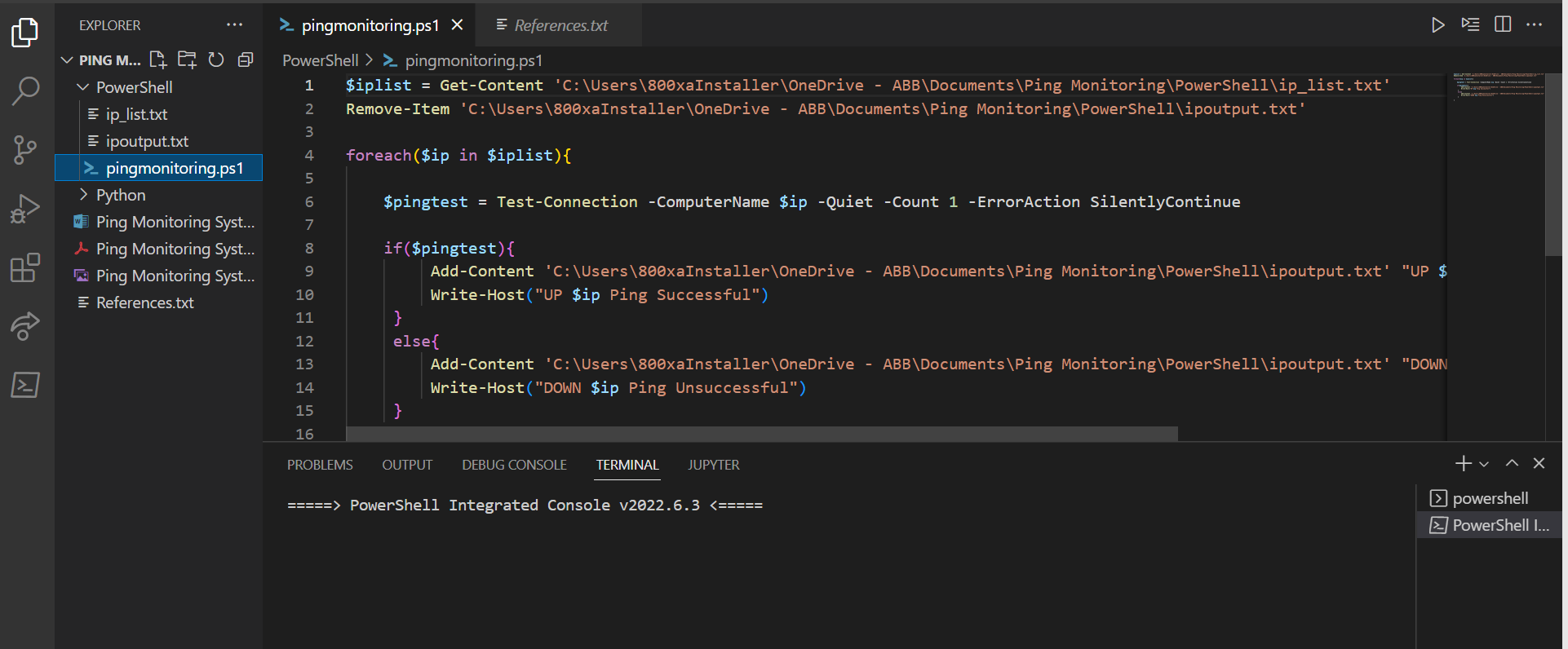
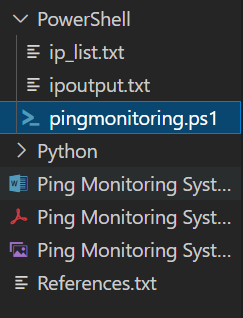
3. Output Text File (ipoutput.txt)

The PowerShell Script will extract IP addresses from the input text file and ping to each of the IP addresses in this list. The ping status will be output to the output text file (This is also the file that Microsoft Power Automate pulls from OneDrive Business).

The PowerShell Script code documentation can be found at this [GitLab Link](https://codebits.abb.com/abb-ping-monitoring-system/abb-ping-monitoring-system/-/tree/main/PowerShell).

### PowerShell Editor Installation [Optional Download]

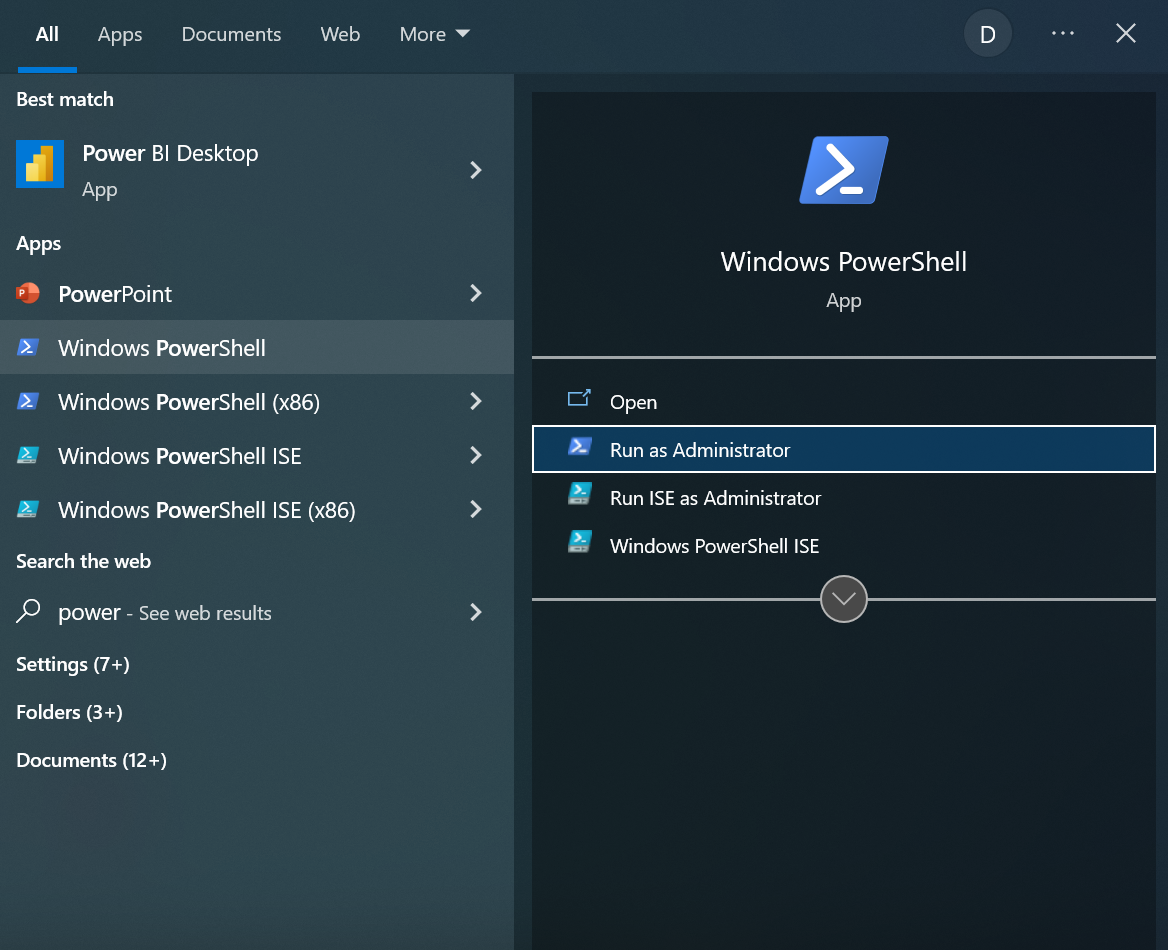
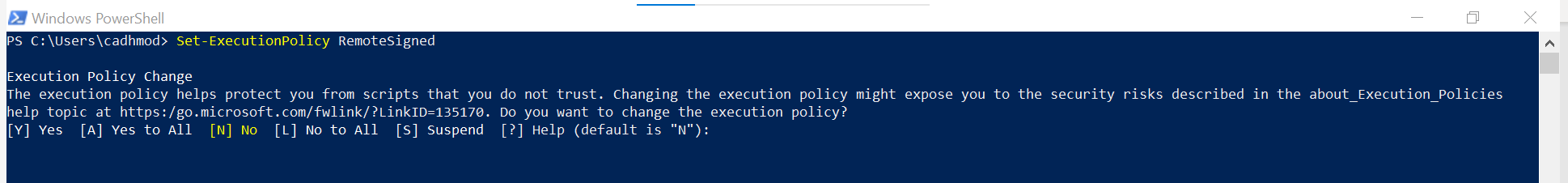
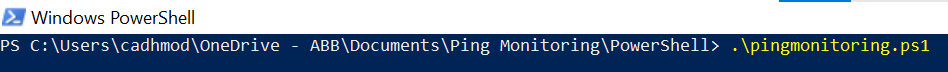
If you would like to edit the PowerShell Script, please follow this step to download Visual Studio Code [Note: Any code editor will work to edit the code including Notepad. Notepad is a very simple editor, already part of Windows OS. Other editors will have more features than Notepad that could make the editing easier and more efficient]

1. Click on the provided Link and download the Windows Version of Visual Studio Code [Big Blue Button]: [Download Visual Studio Code - Mac, Linux, Windows](https://code.visualstudio.com/download)
2. Open Visual Studio Code 
3. Click on “File” on the top left corner 
4. Click on “Open Folder” – Visual Studio Code Keyboard Shortcut for “Open Folder” [Ctrl+K Ctrl+O]
5. Find the Ping Monitoring System Folder, the Folder name should be “Ping Monitoring” 
6. Select the Folder 
7. This will load all the files in the Ping Monitoring Folder to Visual Studio Code 
8. In the Left section of Visual Studio Code, we can see the Python file [pingmonitoring.py], Input File [ip\_list.txt], Output File [ipoutput.txt], and other files
9. Click on the pingmonitoring.ps1 to edit the PowerShell Script, any other files can also be edited [depending on the file type and file can viewed, double click to edit, and all files can be viewed with one click]

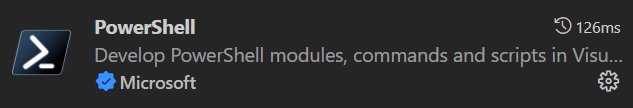
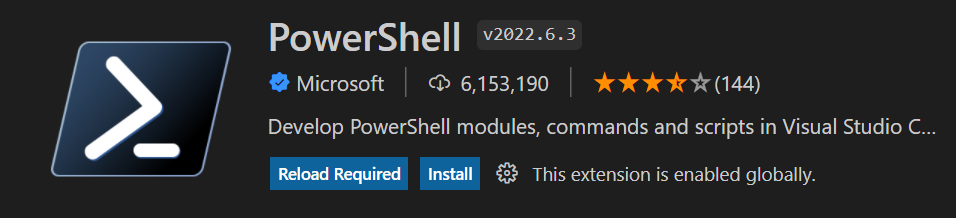
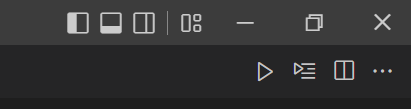
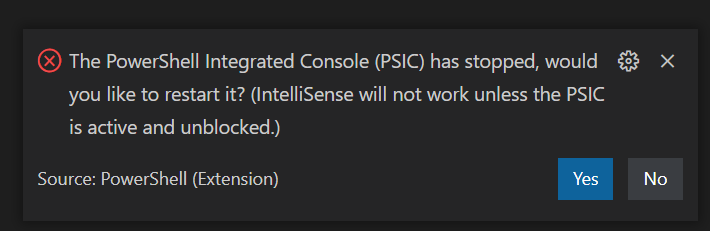
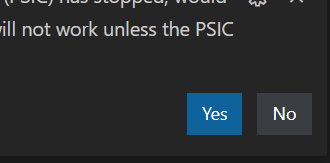
### Running the PowerShell Script

Please follow the steps to run the PowerShell Script manually without the Windows Task Scheduler

#### Running PowerShell Script from PowerShell

1. Open PowerShell as an Administrator, click on “Run as Administrator” 
2. Set the Execution Policy to “RemoteSigned” to safely run the Script 
3. Change the directory to the PowerShell Script with “cd” command [Add double quotes if spaces in the path, for example, name of folder 
4. To run the Script, follow this syntax/structure, “./filename.ps1” 
5. The Script will run in the PowerShell

#### Running PowerShell Script Visual Studio Code

1. Click on the Extension Icon 
2. Search PowerShell in the search box and click on the extension
3. Click on “Install” 
4. Go back to PowerShell Script and click on the run button on the top left, this will run the Script 
   1. The extension can be stopped by clicking on bin button at bottom 
      1. Note: An error can appear when the extension is closed/stopped by clicking on the bin icon 
      2. To solve the error, click on the “Yes” button 
      3. If the “No” button is clicked the extension will not work and we cannot run the PowerShell Script.
         1. Restart the Visual Studio Code and load the Ping Monitoring Again [Visual Studio Code will automatically load the folders and files]

## Python Script

The Python Script section has three parts:

1. Python Script (pingmonitoring.py)
2. Input Text File (ip\_list.txt)
3. Output Text File (ipoutput.txt)

The Python Script will extract IP addresses from the Input Text File and ping to each of the IP addresses in this List. The ping status will be output to the Output Text File (This is also the file that Microsoft Power Automate Pulls from OneDrive Business.

The Python Script code documentation can be found at this [GitLab Link](https://codebits.abb.com/abb-ping-monitoring-system/abb-ping-monitoring-system/-/tree/main/Python).

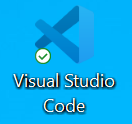
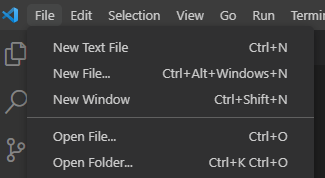
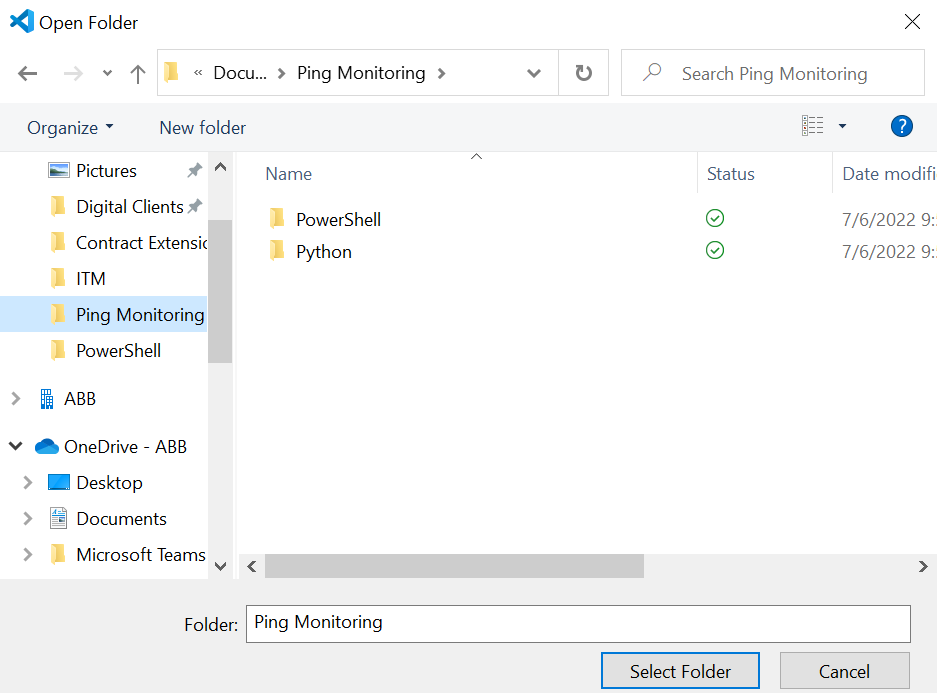
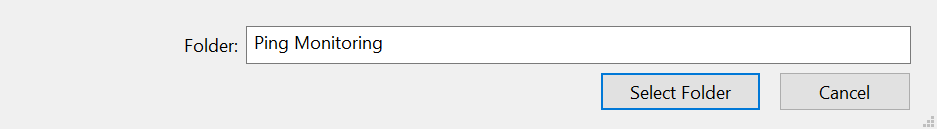
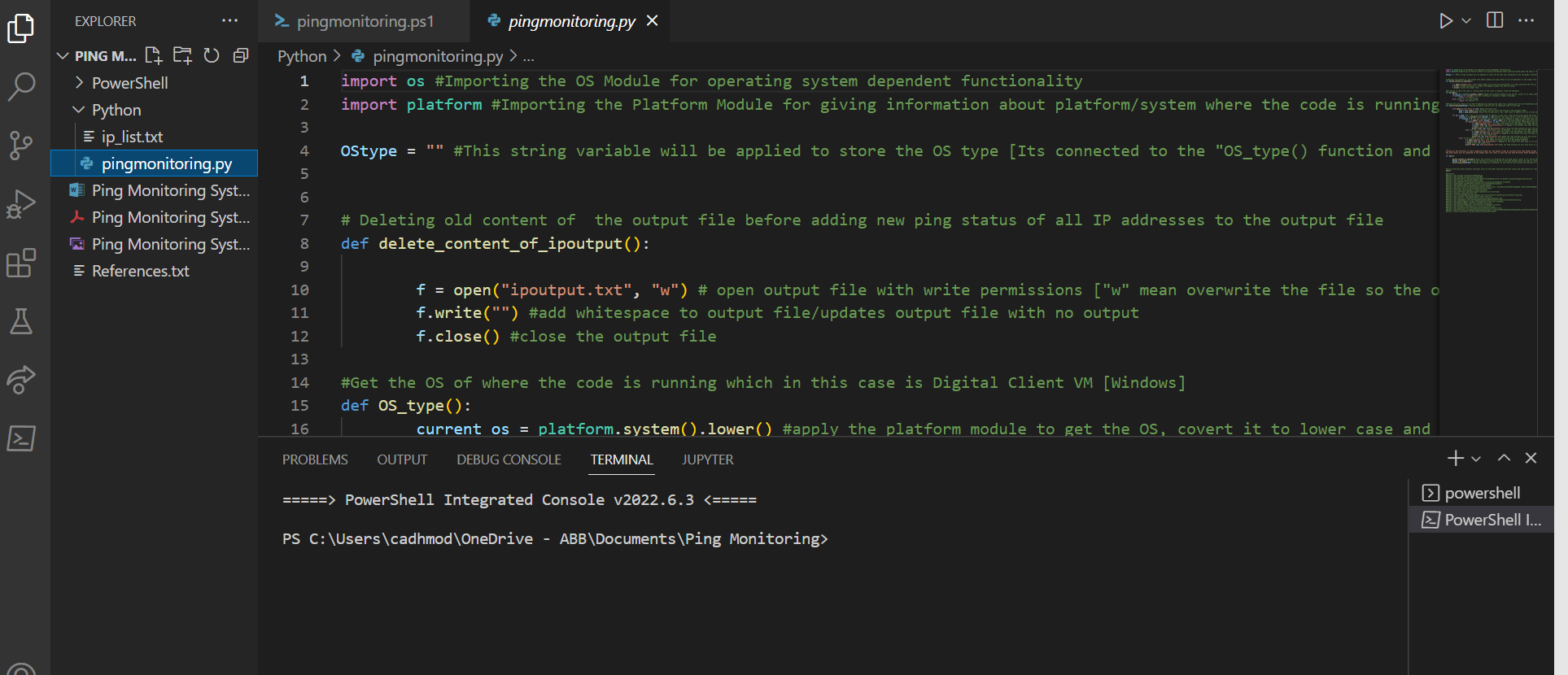
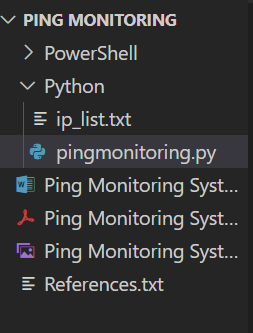
### Python Installation [Without Code Editor]

Python can also be installed without a code editor. We have three options each for different operating system to install Python alone. **We do highly recommend “Python and Code Editor Download” steps to install and configure the Ping Monitoring System instead of the solo Python Installation because it is much easier and more efficient to edit/update the code or even observe the code in Code Editor environment.**

* Windows - [How to Install Python on Windows (networkdirection.net)](https://networkdirection.net/python/intro/overview/install-windows/)
* macOS - [Tutorial: Installing Python on Mac – Dataquest](https://www.dataquest.io/blog/installing-python-on-mac/#:~:text=Install%20Python%203%20with%20the%20Official%20Installer&text=First%2C%20download%20an%20installer%20package,Python%20installer%20on%20your%20Mac.)
* Linux - [How to install Python on Linux | Opensource.com](https://opensource.com/article/20/4/install-python-linux)

### Python and Code Editor Download

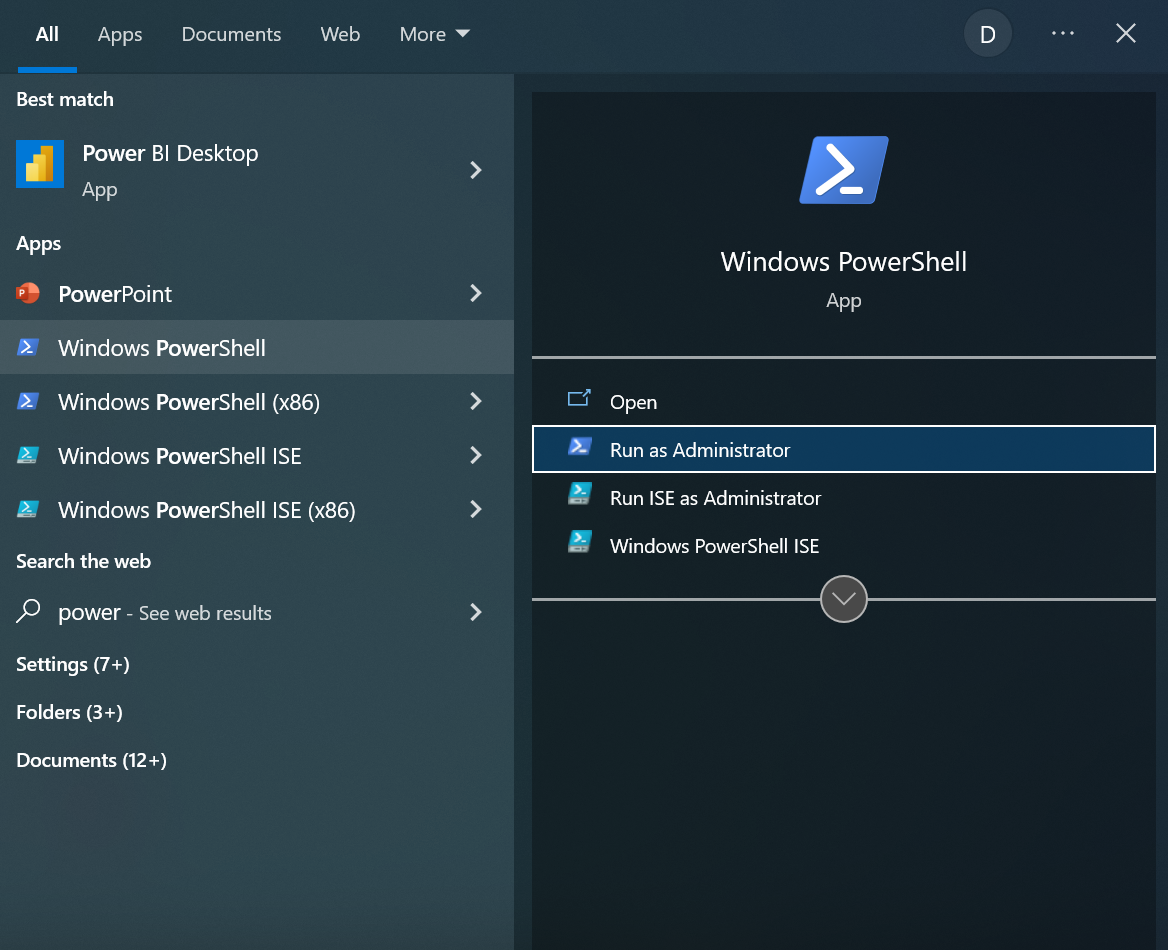
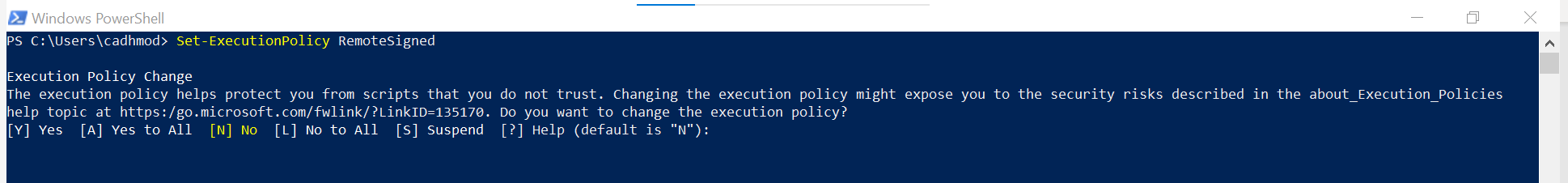
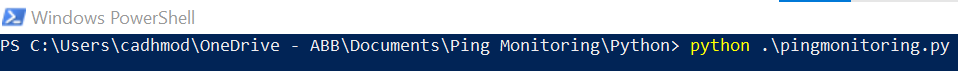
If you would like to edit the Python Script, please follow this step to download Visual Studio Code [Note: Any code editor will work to edit the code including Notepad. Notepad is a very simple editor already part of Windows OS. Other editors will have more features than Notepad that could make the editing easier and more efficient]

1. Download the Visual Studio Code and Python Pack from this link: [https://code.visualstudio.com/docs/Python/coding-pack-Python](https://code.visualstudio.com/docs/python/coding-pack-python)
2. Open Visual Studio Code 
3. Click on “File” on the top left corner 
4. Click on “Open Folder” – Visual Studio Code Keyboard Shortcut for “Open Folder” [Ctrl+K Ctrl+O]
5. Find the Ping Monitoring System Folder, the Folder name should be “Ping Monitoring”
6. Select the Folder 
7. This will load all the files in the Ping Monitoring Folder to Visual Studio Code 
8. In the Left section of Visual Studio Code, we can see the Python file [pingmonitoring.py], Input File [ip\_list.txt], Output File [ipoutput.txt], and other files 
9. Click on the pingmonitoring.py to edit the Python Script, any other files can also be edited [depending on the file type and file can viewed, double click to edit, and all files can view with one click]

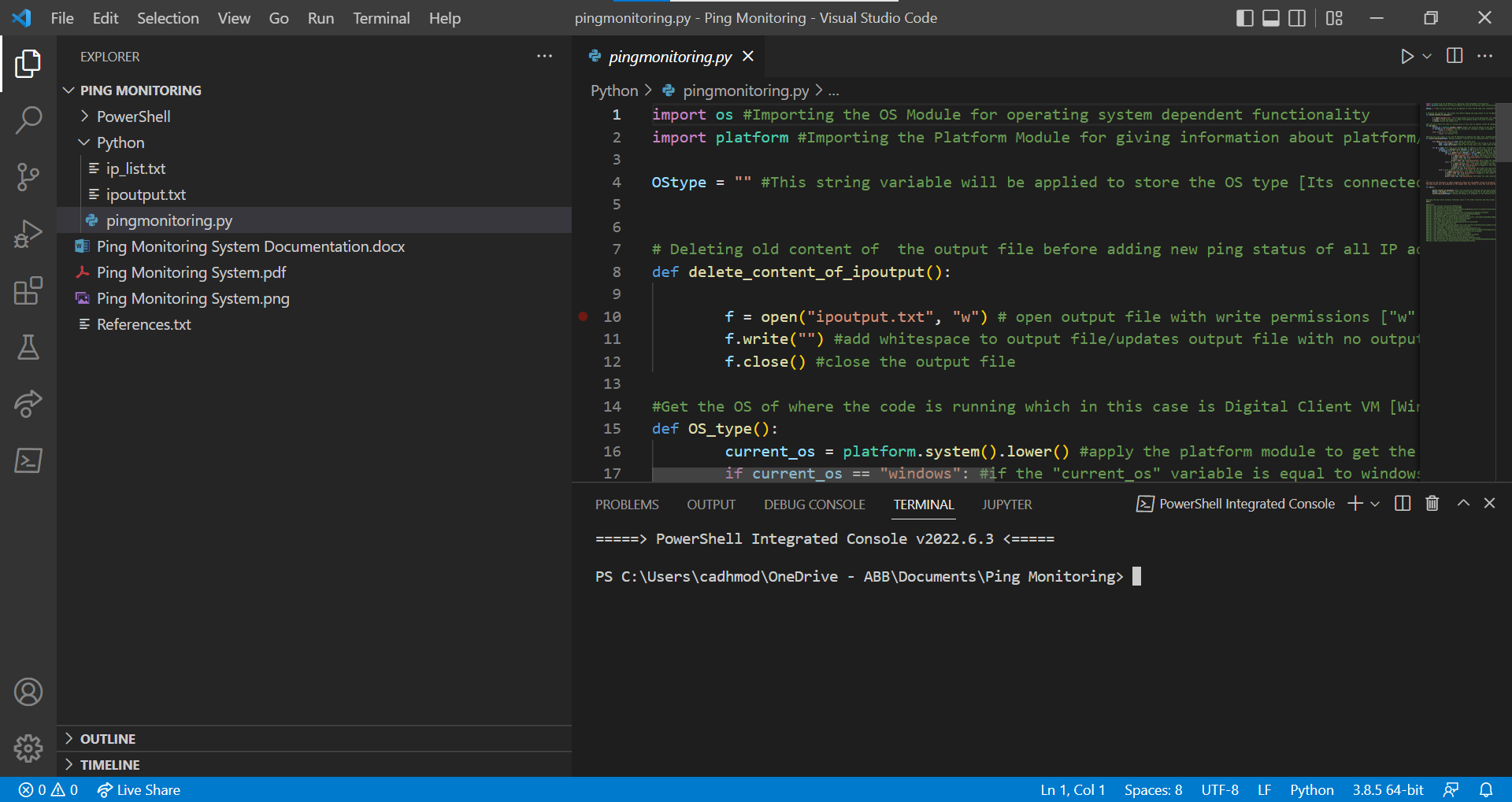
### Running the Python Script

Please follow the steps to run the Python Script without the Windows Task Scheduler

#### Running Python Script from PowerShell

1. Open PowerShell as an Administrator, click on “Run as Administrator” 
2. Set the Execution Policy to “RemoteSigned” to safely run the Script 
3. Change the directory to the PowerShell Script with “cd” command [Add double quotes if spaces in the path, for example, name of folder 
4. To run the Script, follow this syntax/structure, “Python ./pingmonitoring.py” 
5. The Script will run in the PowerShell

#### Running Python Script from Visual Studio Code

1. Load the “Ping Monitoring” into Visual Studio Code and open the pingmonitoring.py 
2. Click on the Run Button [Triangle] at the top right corner of the Visual Studio Code  
3. This will run the Python Script in the terminal at botom of Visual Studio Code

# OneDrive Business

The OneDrive Business should be downloaded to the host machine because the Power Automate will pull the Output File from the OneDrive Business. For this to occur, the PowerShell Script and Python Script and its files must be stored and put in OneDrive Business, this will also keep the files in the cloud/internet and the ABB Employees can easily access them.

This will also allow edits to be done on more than one computer. For example, if an employee edits the file on their personal laptop, another ABB employee can also access the files after they are updated.

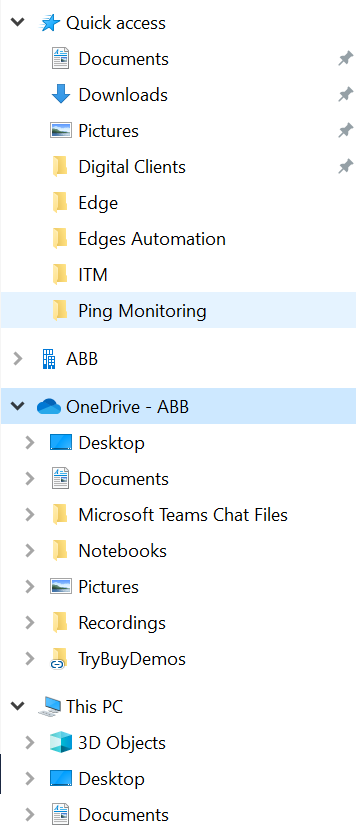
Follow this tutorial to install and configure OneDrive Business [Please don’t’ skip the order of the steps]

1. [https://www.proximus.be/support/en/id\_sfaqs\_oned\_install/self-employed-and-small-companies/support/website-and-cloud/onedrive-for-business/installation-and-configuration/activate-and-install-onedrive-for-business.html#](https://www.proximus.be/support/en/id_sfaqs_oned_install/self-employed-and-small-companies/support/website-and-cloud/onedrive-for-business/installation-and-configuration/activate-and-install-onedrive-for-business.html)
2. <https://www.youtube.com/watch?v=2BfflozfvNE>

### Important Note

From the information provided above on why OneDrive Business is being applied, the Ping Monitoring System Folder must be in a location on your machine where OneDrive business can sync the files:

Follow this Steps to create or move the folder to a location where the Files and Folder can be Synced to OneDrive Business [Note: There many ways to achieve this task, these steps show one of the ways in achieving this task]

1. Open File Explorer 
2. Click on the OneDrive Option, part of the Left Section 
3. Select Location, where it will be easy for you to find the Folder/Files for example:
4. Drag the Folder/Files into the Location or Create Folder/Files in the Location

# Windows Task Scheduler

The Windows Task Scheduler has two tasks:

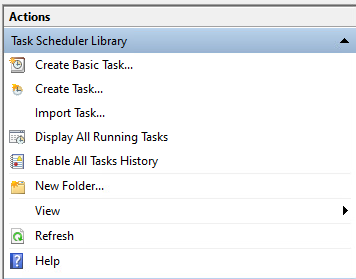
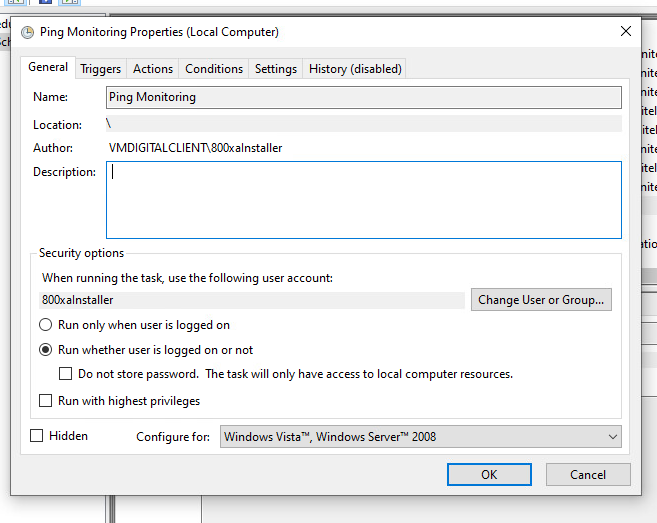
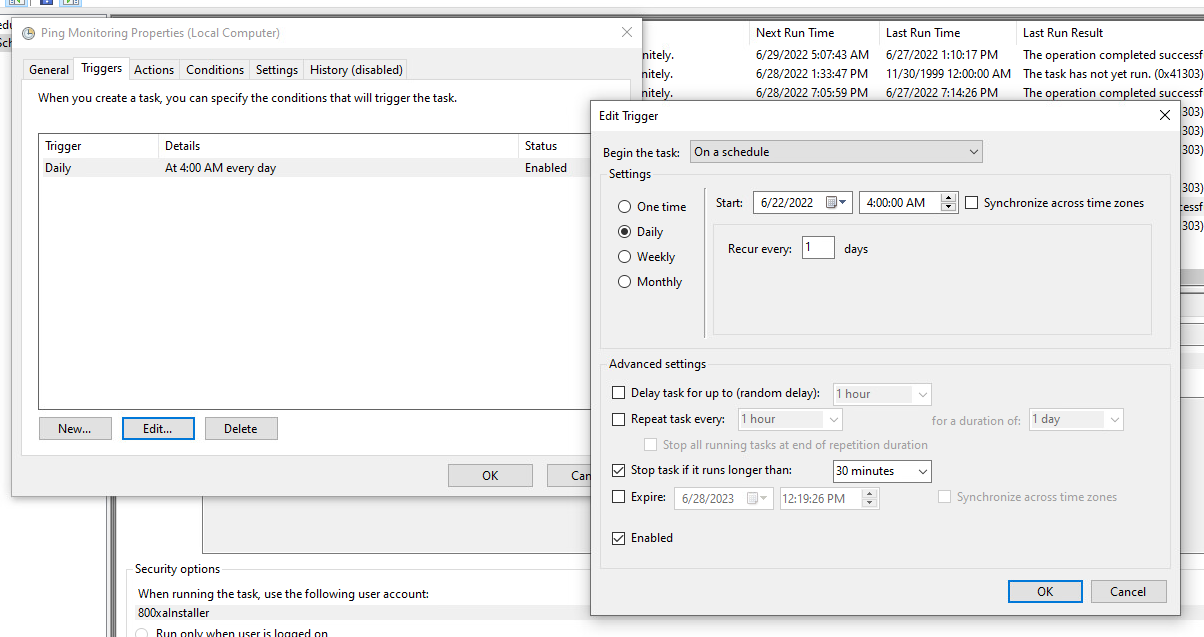
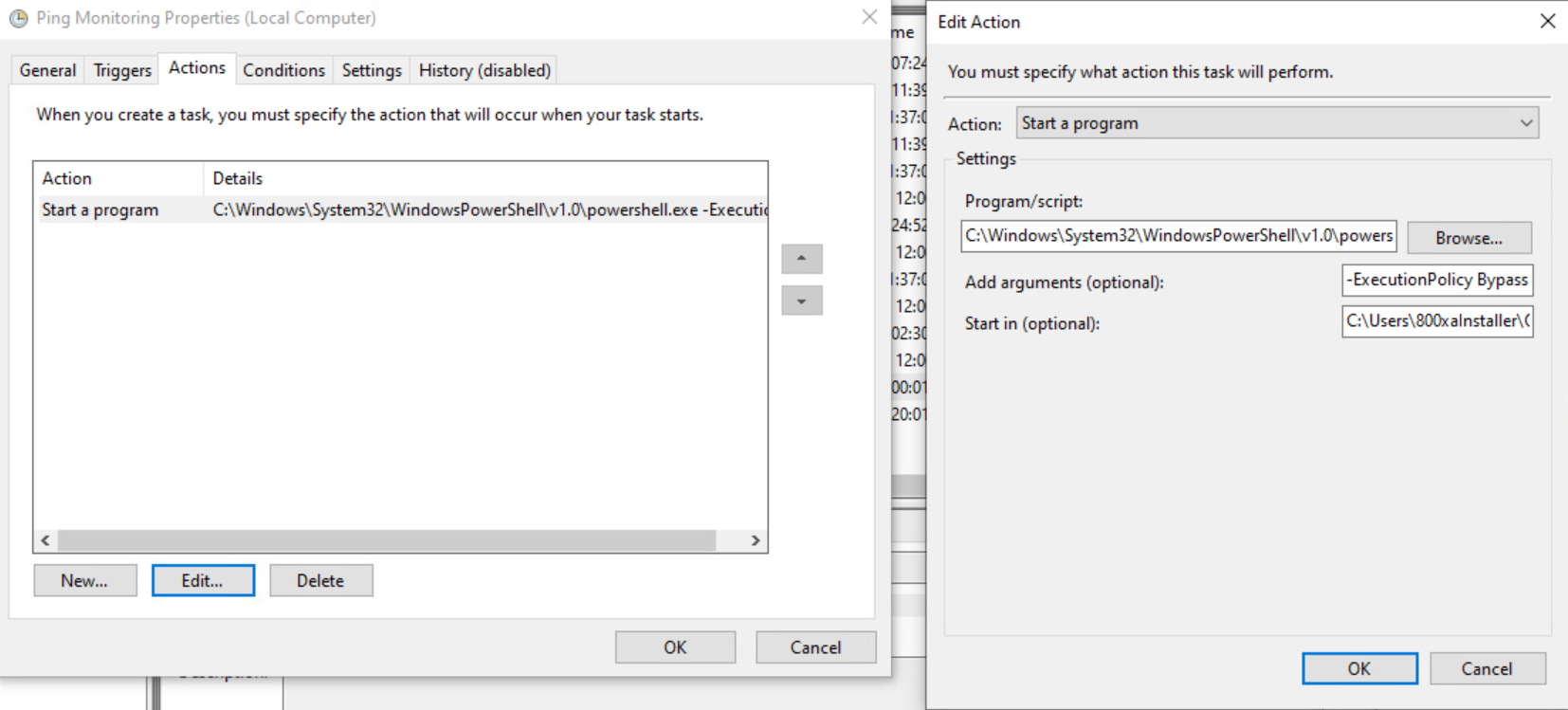
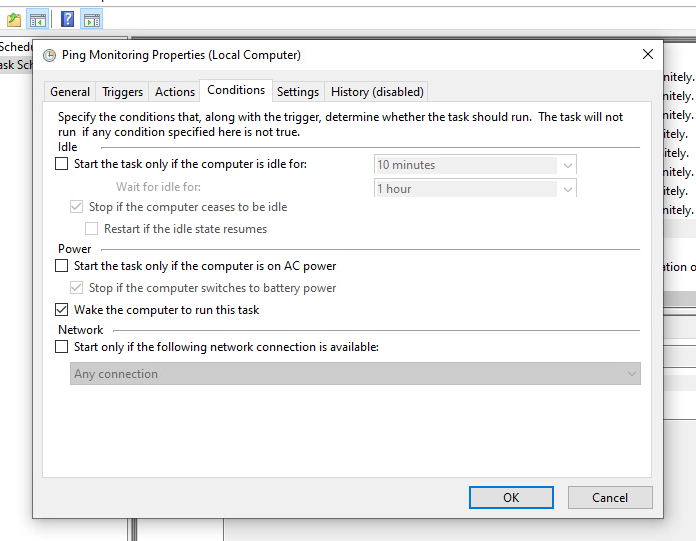
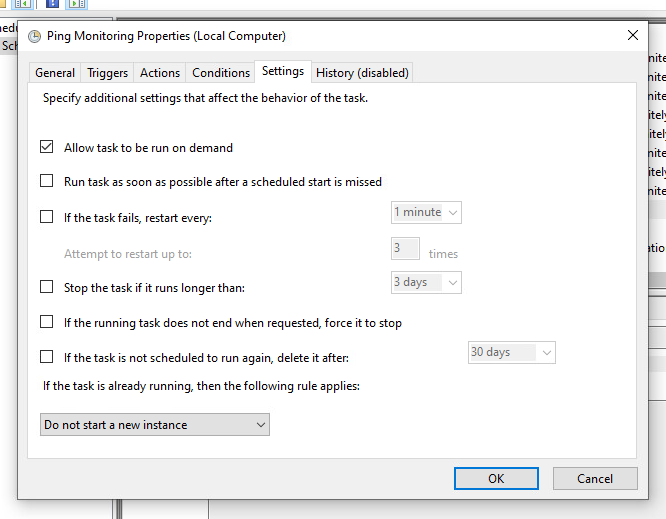
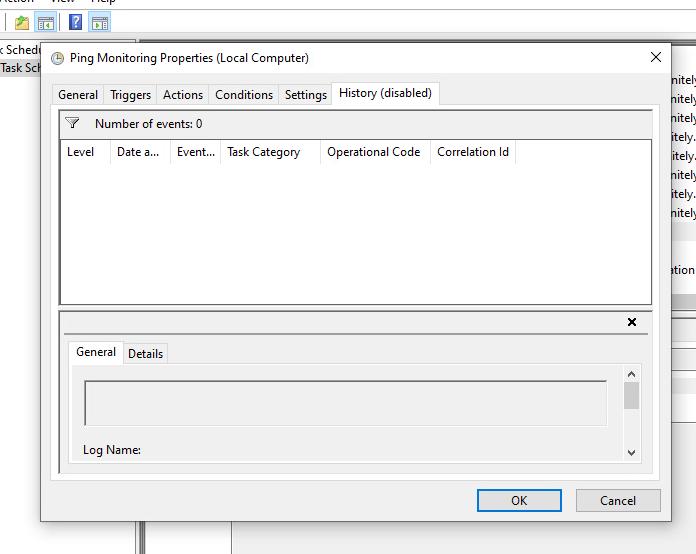
1. Automate the running of the PowerShell Script/Python Script
2. Automate the OneDrive Syncing

## PowerShell Script

The Digital Client VM task scheduler will contain task for automating the running of the PowerShell Script. The task that automates the running of the PowerShell Script is in the below screenshot.



The steps to create this task are:

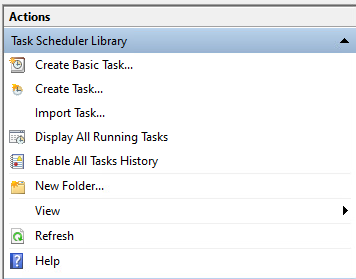
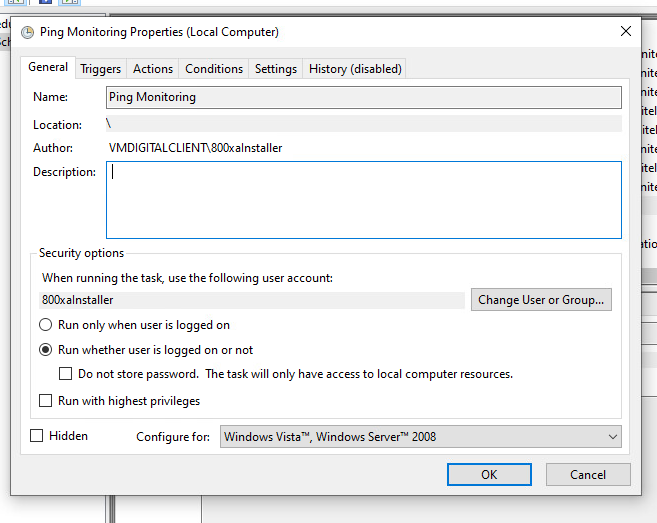
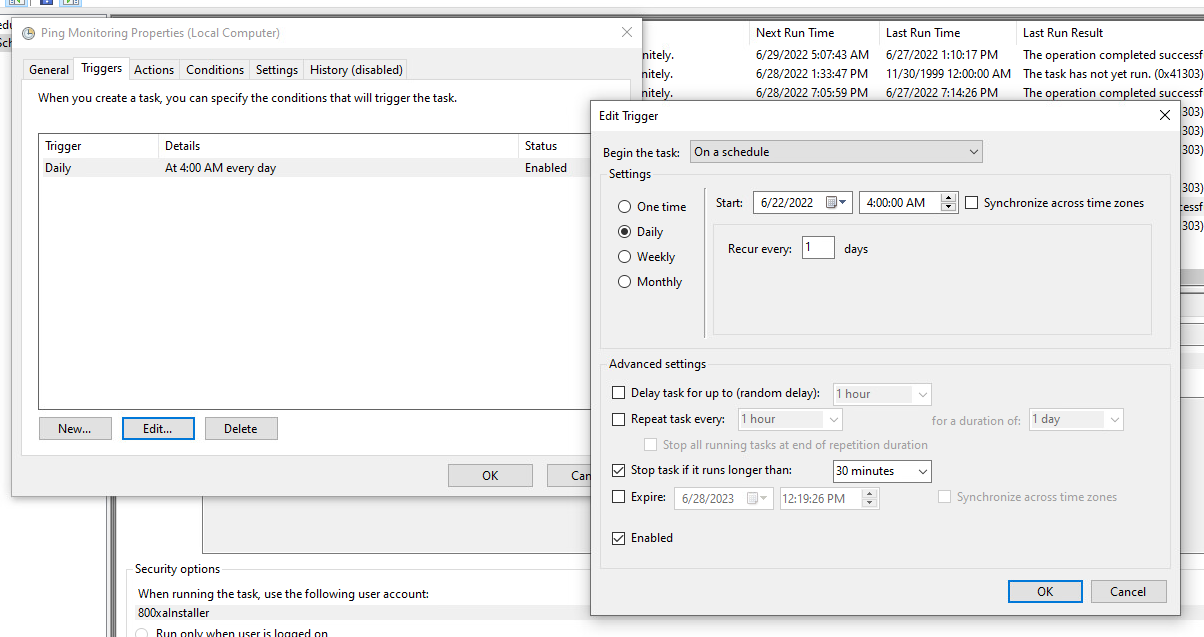
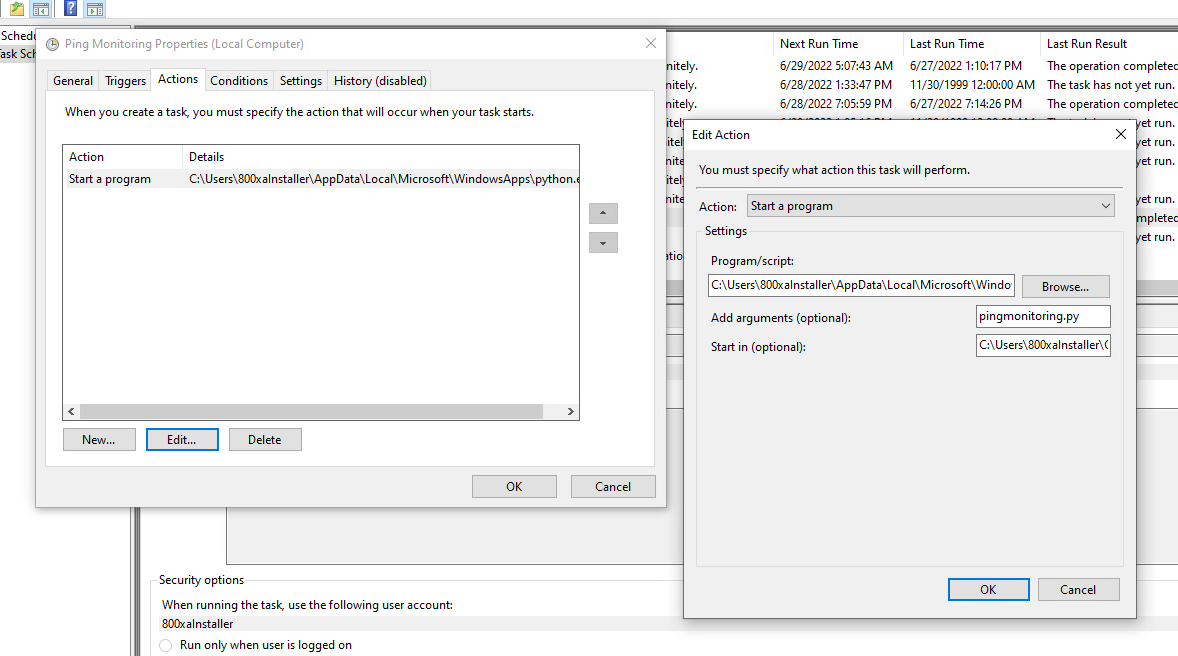
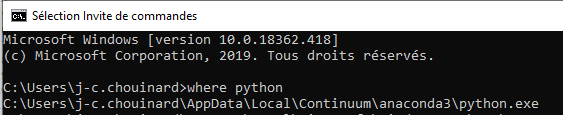
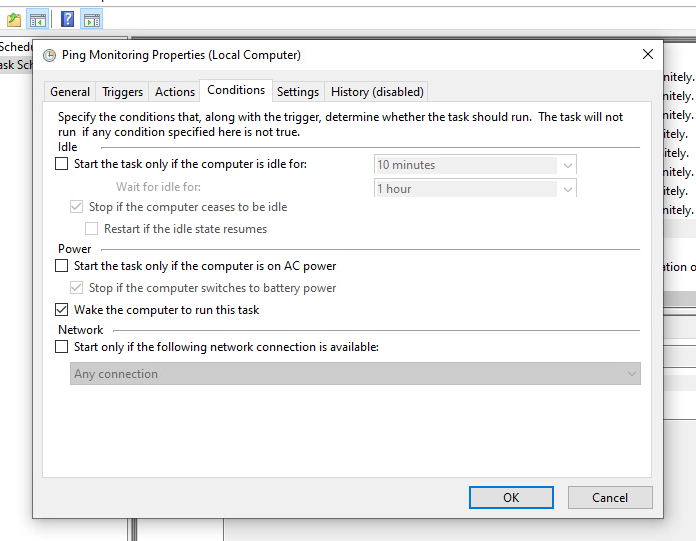
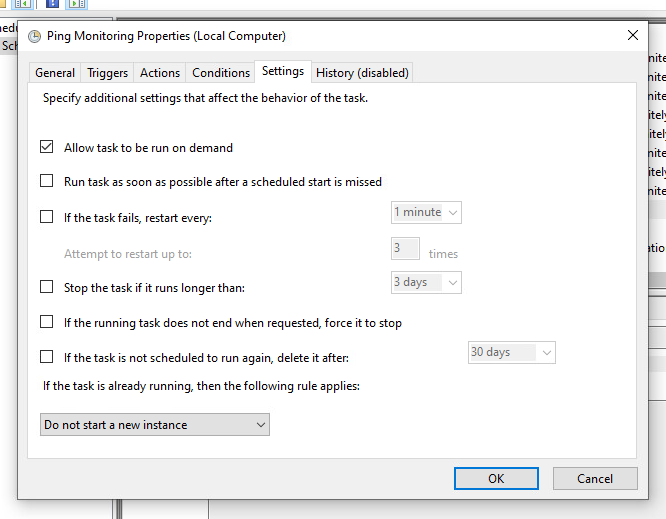
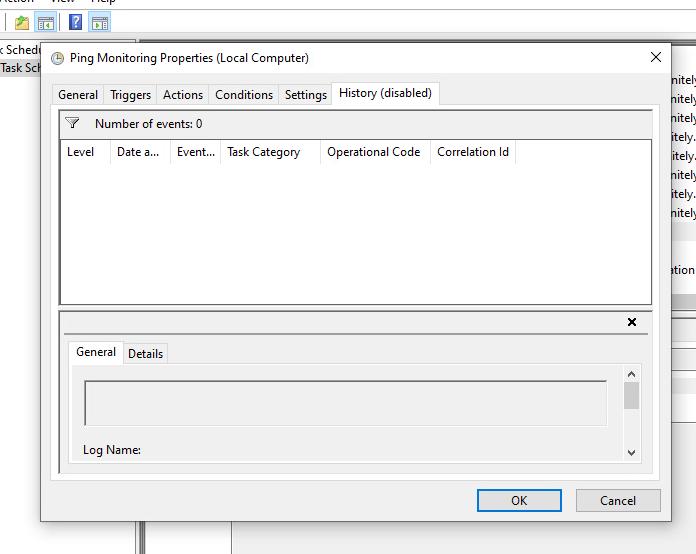
1. Click on Create Task to initiate the task creation 
2. Fill in the information according to the below screenshot 
3. Click on “Trigger” tab and then click on “New”, and follow the screenshot, click “OK” 
4. Click on “Actions” tab and then click on “New”
   1. Select “Start a program”
   2. For Program/Script, click on Browse
      1. Add “PowerShell.exe” located at - C:\Windows\System32\WindowsPowerShell\v1.0 [Include both path and the PowerShell.exe, look at sub step a]
         1. C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell.exe
   3. For “Add arguments (optional)” and “Start in (optional)”
      1. Add arguments: -ExecutionPolicy RemoteSigned -File “C:\Path\Scriptname.ps1”
         1. For Example: C:\Users\cadhmod\OneDrive - ABB\Documents\Ping Monitoring\PowerShell\pingmonitoring.ps1
      2. Start in (Optional): C:\path [no slash at the end of the path]
         1. For Example: C:\Users\cadhmod\OneDrive - ABB\Documents\Ping Monitoring\PowerShell
   4. Click “OK”
5. Click on “Conditions” tab, and select the options from the below screenshot 
6. Click on “Settings”, select the options 
7. Do not change anything for “History (disabled)” 

## Python Script

The Digital Client VM task scheduler will contain task for automating the running of the Python Script. The task that automates the running of the Python Script is in the below screenshot.



The steps to create this task are:

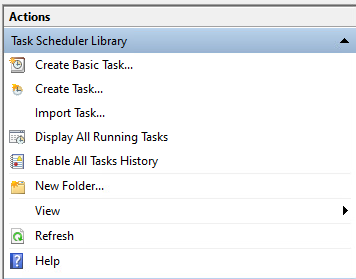
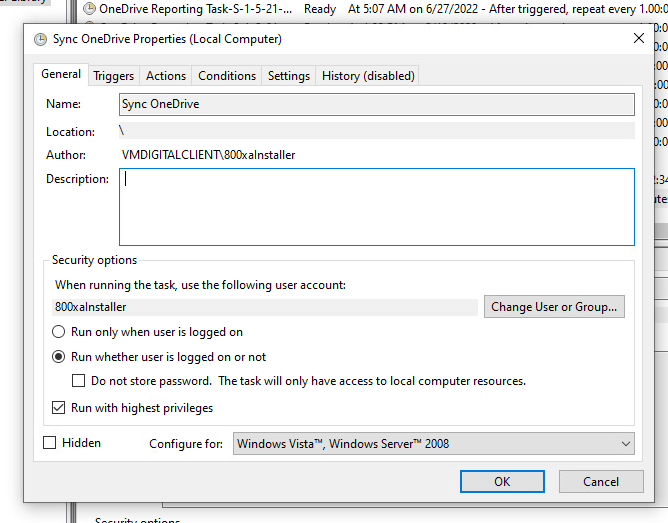
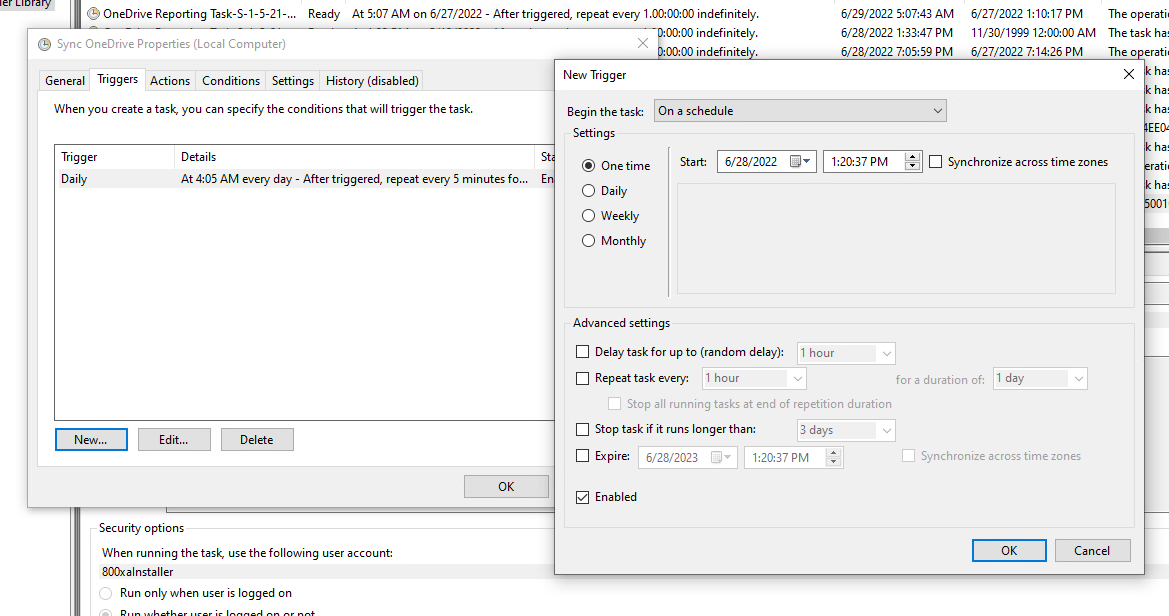
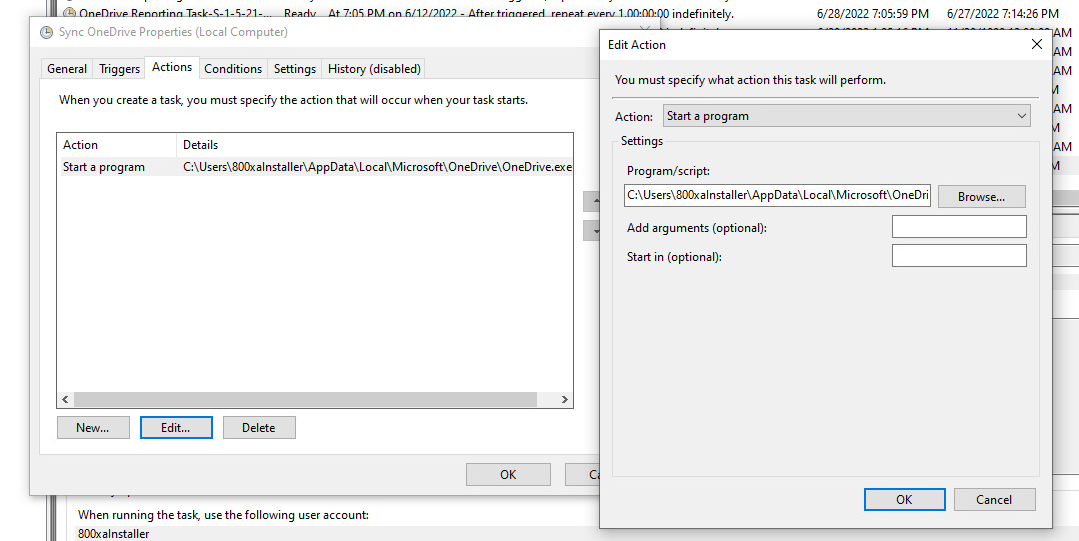
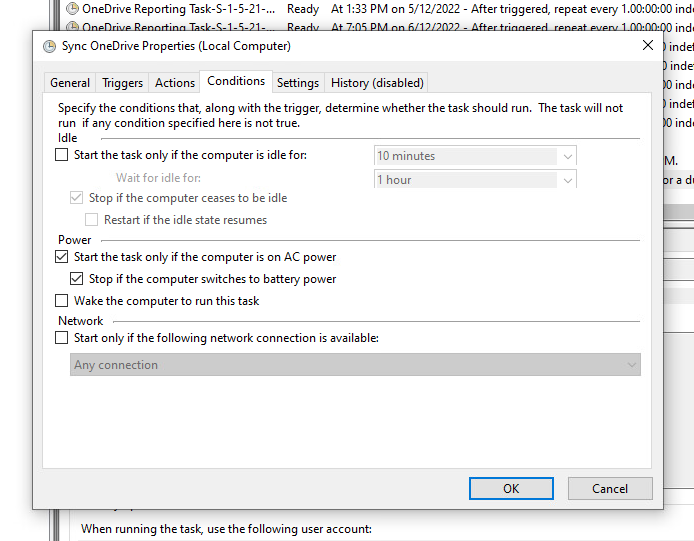
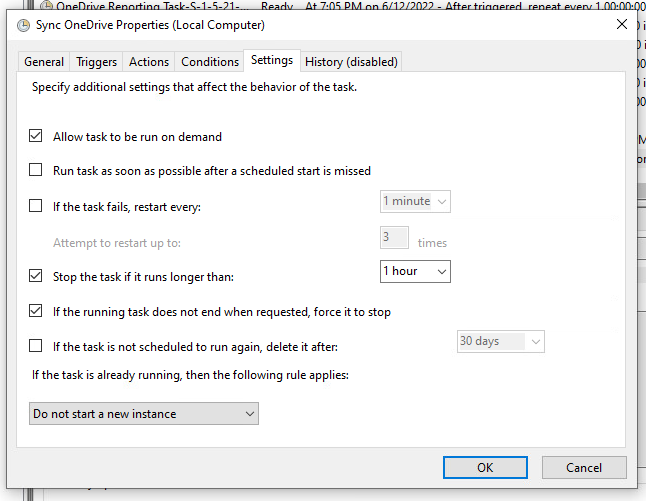
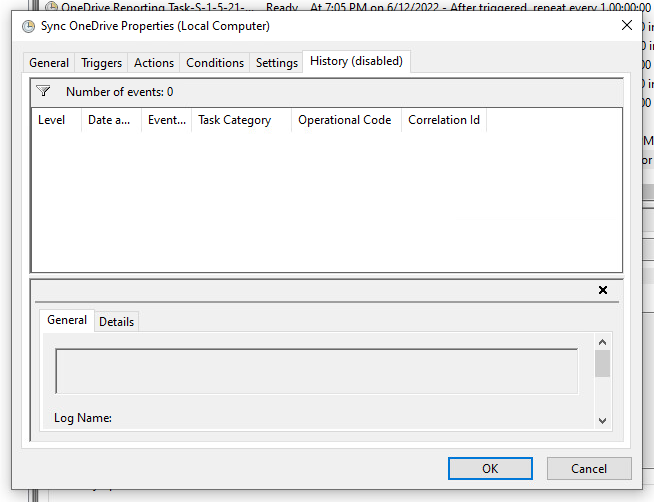
1. Click on Create Task to initiate the task creation 
2. Fill in the information according to the below screenshot 
3. Click on “Trigger” tab and then click on “New”, and follow the screenshot, click “OK”
4. Click on “Actions” tab and then click on “New”
   1. Select “Start a program”
   2. For Program/Script
      1. Run “Command Prompt” and “Widows PowerShell” as administrator to find the Python path
      2. Enter “where command”, this will present the path of the Python Executable file 
         1. For Example: C:\yourpath\Python.exe
   3. For “Add arguments (optional)” and “Start in (optional)”
      1. “Add arguments (optional): Add the Python file name
      2. Start in (optional): Add the path to your Python file
         1. For Example: C:\user\your\_Python\_project\_path [Don’t add the filename, only add where the file is located]
   4. Click “OK”
5. Click on “Conditions” tab, and select the options from the below screenshot
6. Click on “Settings”, select the options 
7. Do not change anything for “History (disabled)” 

## OneDrive Business

The “Sync OneDrive” task allows the Ping Monitoring System files to be updated even when the user is not signed in or is signed into another account [signed in as another user].



The steps to create this task are:

1. Click on Create Task to initiate the task creation 
2. Fill in the information according to the below screenshot 
3. Click on “Trigger” tab and then click on “New”, and follow the screenshot, click “OK” 
4. Click on “Actions” tab and then click on “New”
   1. Select “Start a program”
   2. Program/Script
      1. Add the path for OneDrive Executable File, should be similar to: C:\Users\USERNAME\AppData\Local\Microsoft\OneDrive\OneDrive.exe
   3. For “Add arguments (optional)” and “Start in (optional)”, keep it empty
   4. Click “OK”
5. Click on “Conditions” tab, and select the options from the screenshot 
6. Click on “Settings”, select the options 
7. Do not change anything for “History (disabled)” 

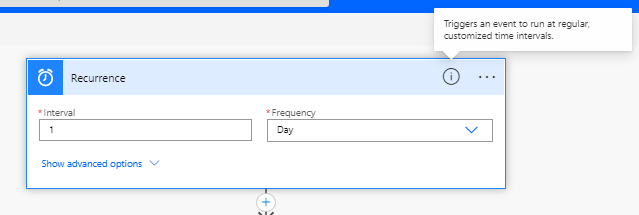
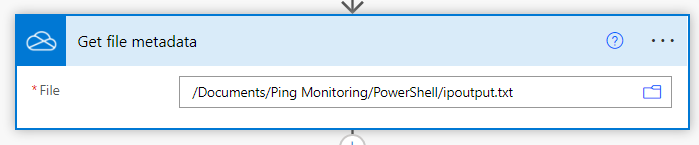
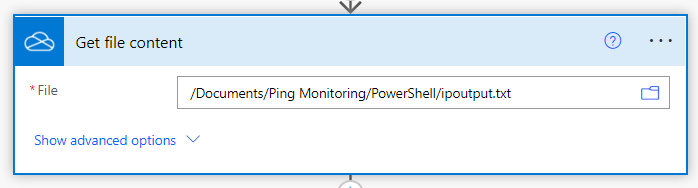
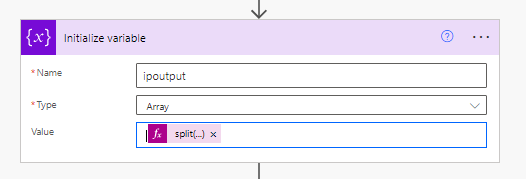
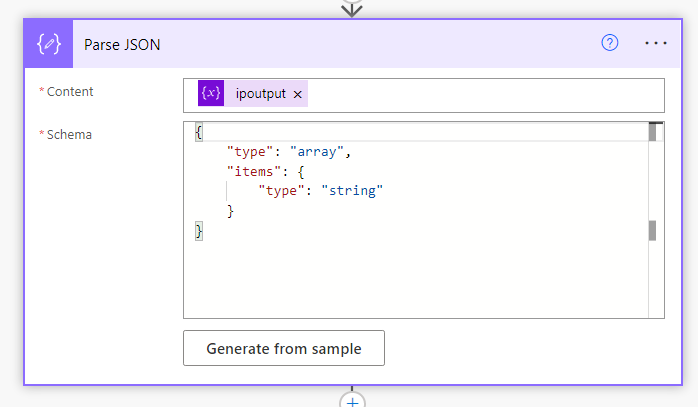
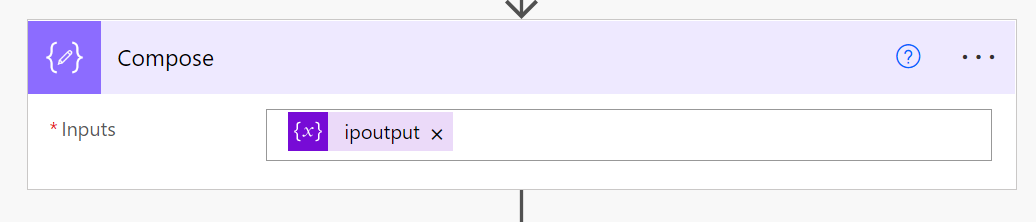
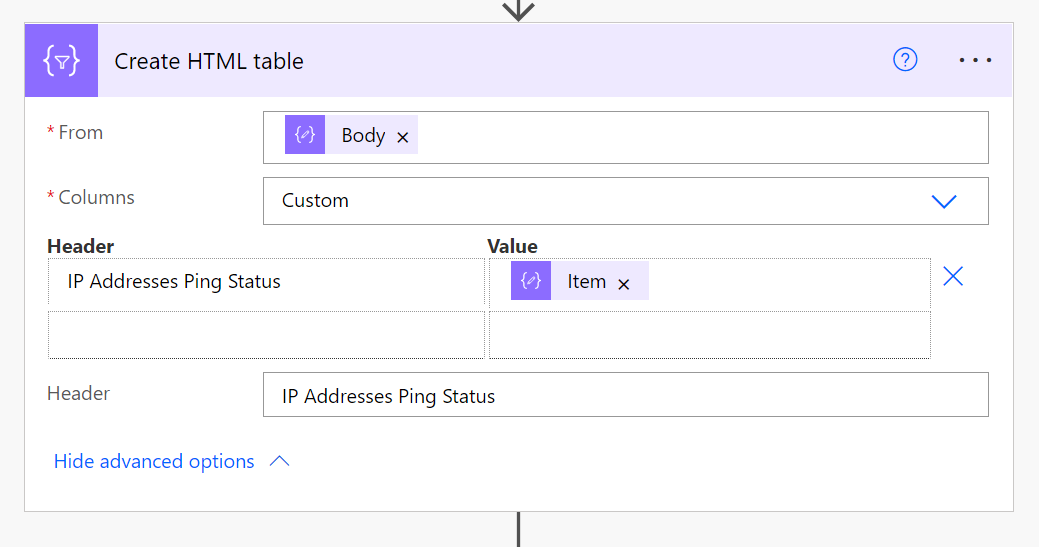
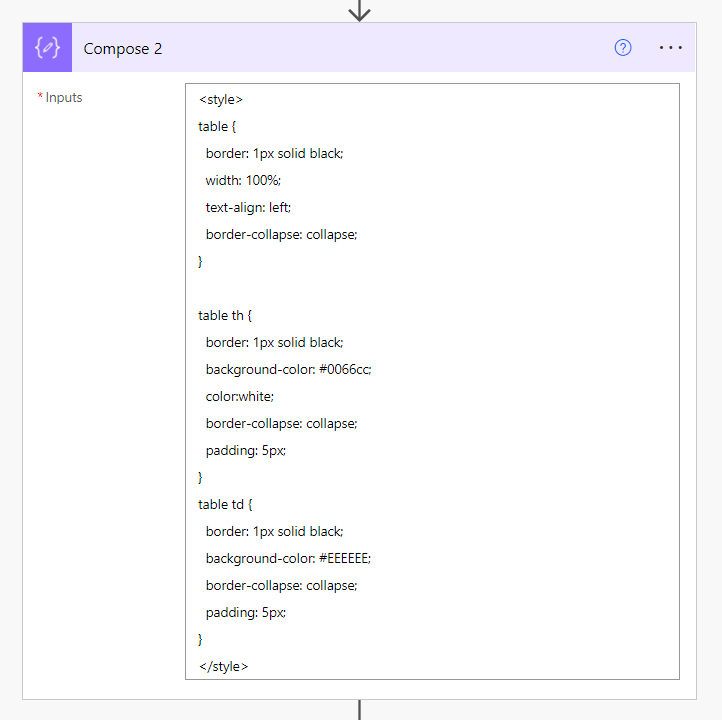
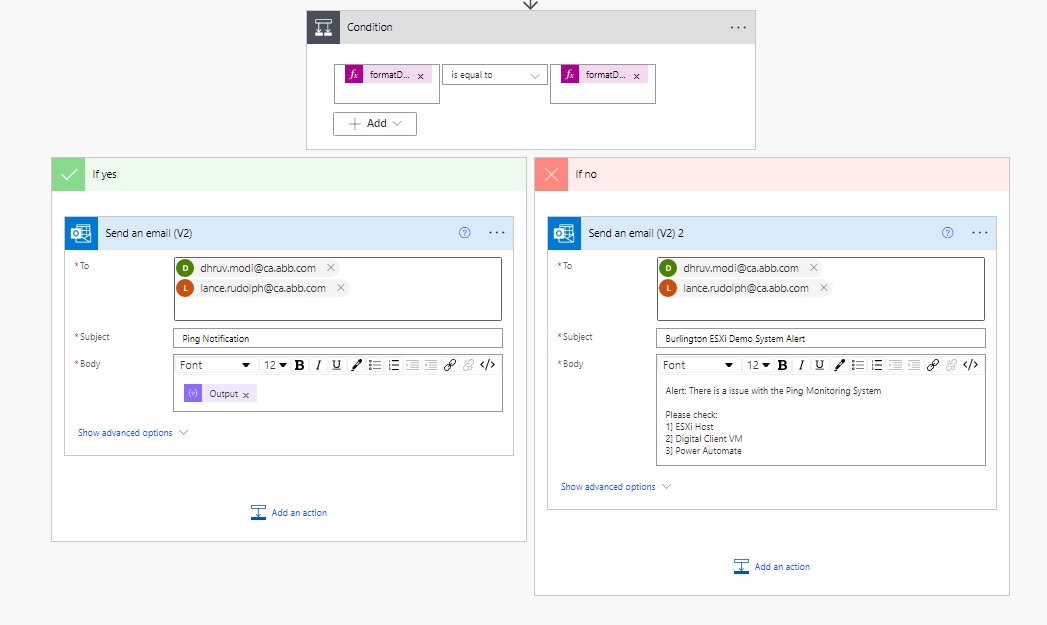
# Microsoft Power Automate

In the Ping Monitoring System, the Microsoft Power Automate will pull the output file and apply it to the flow. The main function of the Power Automate (Microsoft Power Automate) is to send email to ABB Employee/Team which contains Ping Status for the IP Addresses and if there is an issue with system than an email will be sent regarding problems, this will inform the email receiver to check different parts of the Ping Monitoring System.

## Microsoft Power Automate Flow

Summary of the Flow: 

Microsoft Power Automate Flow Steps Details:

1. Initiates the Power Automate Flow at a specific time [Trigger Based] 
2. Retrieve the metadata of the Output File of the PowerShell Script/Python Script [Focus for this step is the time when the file is updated/created, which will be applied in the later steps] 
3. Retrieve the content of the Output File of the PowerShell Script/Python Script 
4. Creating an Array variable for the file content 
   1. Value: split(outputs('Get\_file\_content')?['body'], ',') [This creates an Array of the IP Addresses separating by a comma in the Output File]
5. Converting the Array Output File to a JSON with a specific structure 
6. Creating an object from the variable created in step 4 
7. Create a HTML Table from the step 6 object 
8. Change the HTML Table design 
9. Add a condition Block, this Block will compare the updated/modified date of the Output File with the current date. If the dates are same [that means that the date when output file has been modified /updated matches the current date. This will occur when both the Windows Scheduler Tasks have been executed on that day], Power Automate condition Block will send an email [Left Email] with ping status of all the IP addresses. If the dates are different, then alert email [Right Email] will be sent, which will instruct ABB Employee(s) to check the ESXi host, Digital Client VM, and Power Automate. 
   1. Condition Left Side: formatDateTime(outputs('Get\_file\_metadata')?['body/LastModified'],'yyyy-MM-dd') [This will function/expression will get the last modified date and time of the Output File and format it to only show the Date, for example 2022-06-29]
   2. Condition Right Side: formatDateTime(utcNow(), 'yyyy-MM-dd') [This function/expression will get the current Date in the same format as the condition on the left side]
   3. Left Email “Output” Block in the Body of the Email: This is the output of the HTML Table