COHESITY

How to Run Cohesity PowerShell Scripts

This document covers the items that need to be considered to successfully execute Cohesity PowerShell scripts, both interactively and on a schedule.

Preparing the Workstation or Server

Cohesity PowerShell scripts use the Cohesity REST API to manage a Cohesity cluster. This requires a fairly recent version of PowerShell (5.0 or later) to support modern security protocols (TLS v1.2).

Windows 10 and Windows Server 2016 ship with PowerShell 5.1, but older versions of Windows such as Windows 7/8 or Windows Server 2012r2 ship with older versions of PowerShell that are not acceptable. To upgrade the version of PowerShell, go to https://docs.microsoft.com/en-us/powershell/scripting/install/installing-windows-powershell?view=powershell-6#upgrading-existing-windows-powershell and get the WMF 5.1 upgrade package for the correct OS version (be careful to select the package for Windows 2012 R2, not Windows 2012 if that's what you're after). Note that the upgrade will require a reboot of the server.

What User Will be Running the Script

Logon (or remote desktop) to the server (where the script will be run) as the user account that will run the script. This will allow you to test that the user has the required permissions, and store any secure passwords that will be required for scripts to run unattended.

Downloading the Scripts

Many of the commonly used Cohesity PowerShell scripts are found at https://github.com/bseltz-cohesity/scripts. These scripts are cross-platform and can run on Windows, Linux and Mac. It's important to follow the download instructions carefully, otherwise the scripts could end up with incorrect line endings or even html (from the github website) embedded in the code.

The README for each script contains download instructions in the form of PowerShell commands that can be copied/pasted into PowerShell. These commands download the files and convert the line endings as appropriate to the target OS.

If the download commands throw an error, you may need to launch Internet explorer to answer the first-run security questions, then retry the download commands. If the download commands simply won't work, you can click on each file, then **click the RAW button**, select and copy the contents and paste into a new powershell file (extension .ps1) on the server. Note: if you do not click the raw button, you are copying an HTML rendering of the PowerShell code, not clean PowerShell code.

Test the Script by Hand

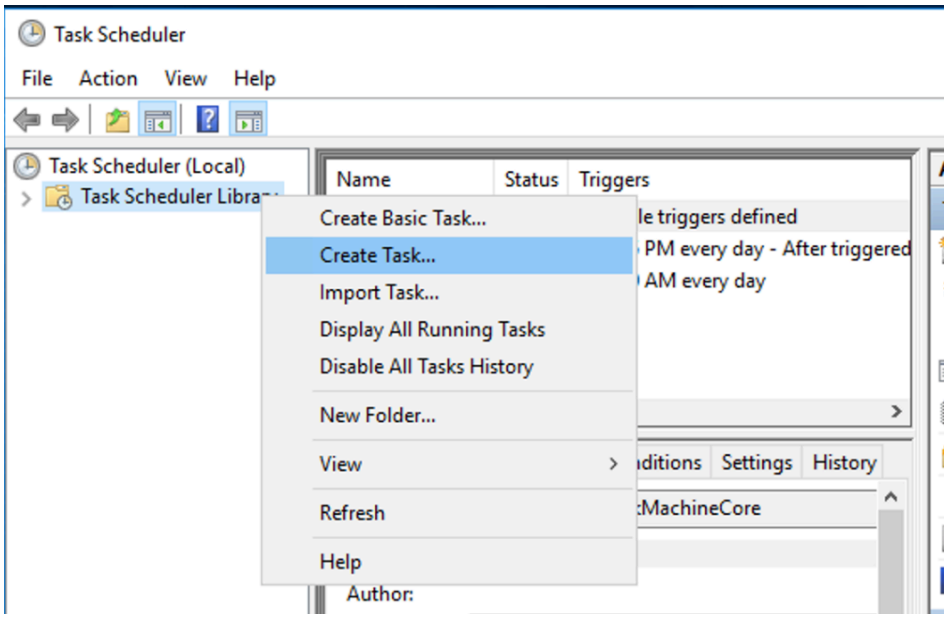
Once the script is in place, test the script by running it by hand. Open a PowerShell window and run the script as desired. For example:

```
.\strikeReport.ps1 -vip mycluster -username myuser -domain mydomain.net -
smtpServer mail.mydomain.net -sendTo myuser@mydomain.net -sendFrom
mycluster@mydomain.net
```

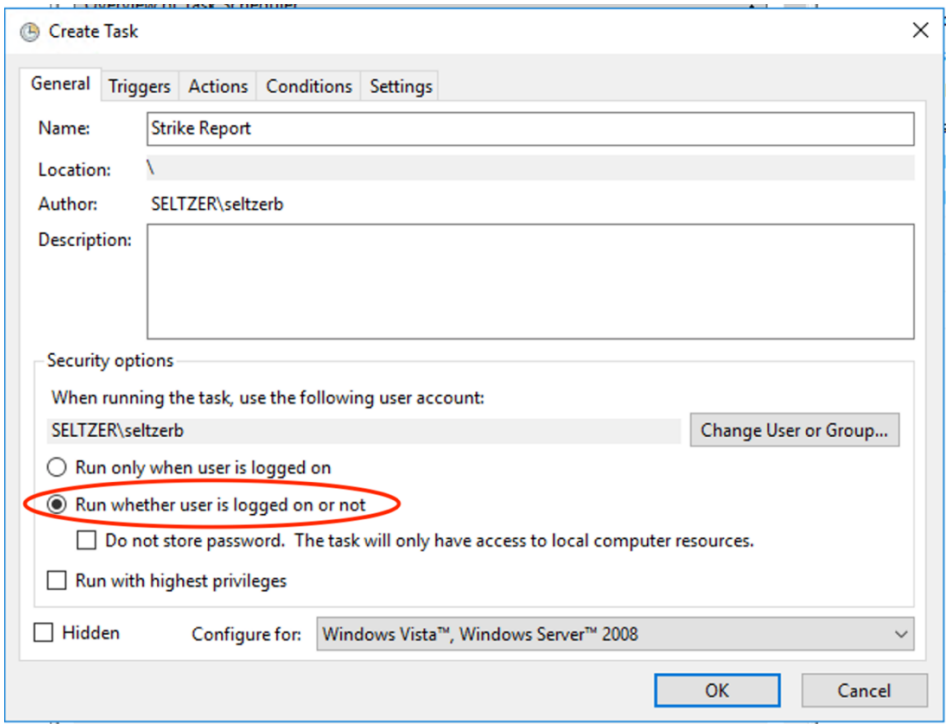
Once the script has run successfully, we can schedule the script to run periodically.

Schedule a PowerShell Script Using Windows Task Scheduler

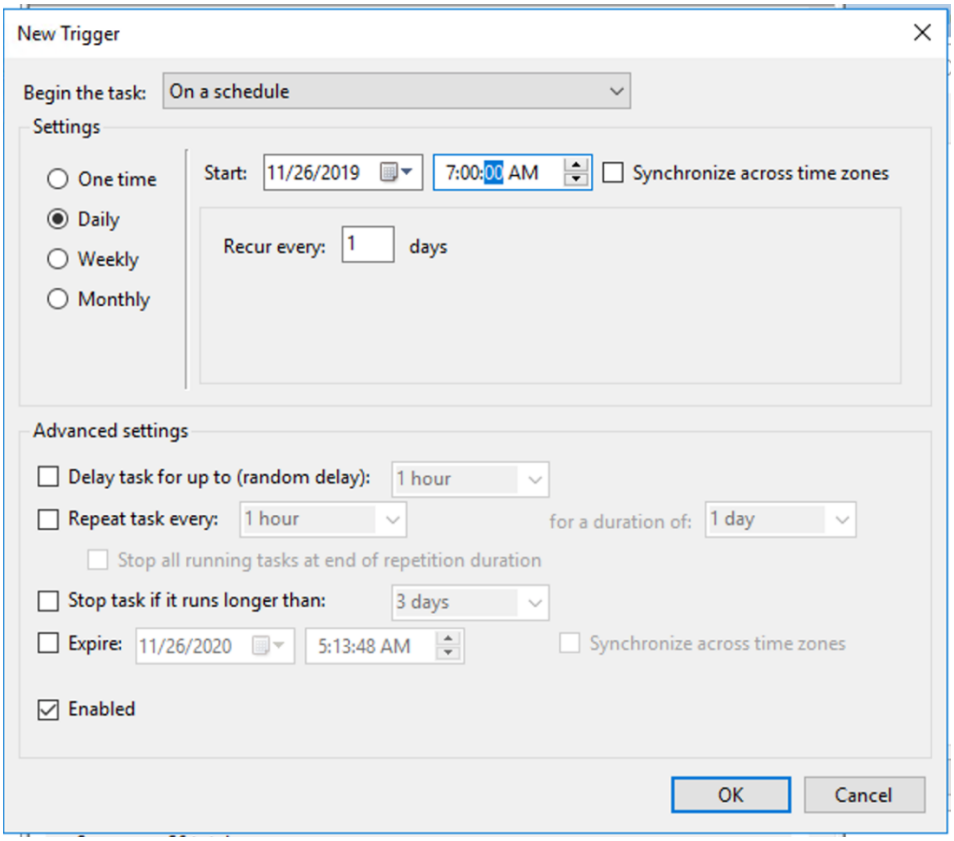
On the server, still logged in as the user who will run the script, launch Task Scheduler. Right-click the Task Scheduler Library folder and click “Create Task”.



On the General tab, give the task a name, and select “Run whether user is logged on or not”.



On the Triggers tab, create a new trigger and specify a schedule to run the script.



On the Actions tab, specify:

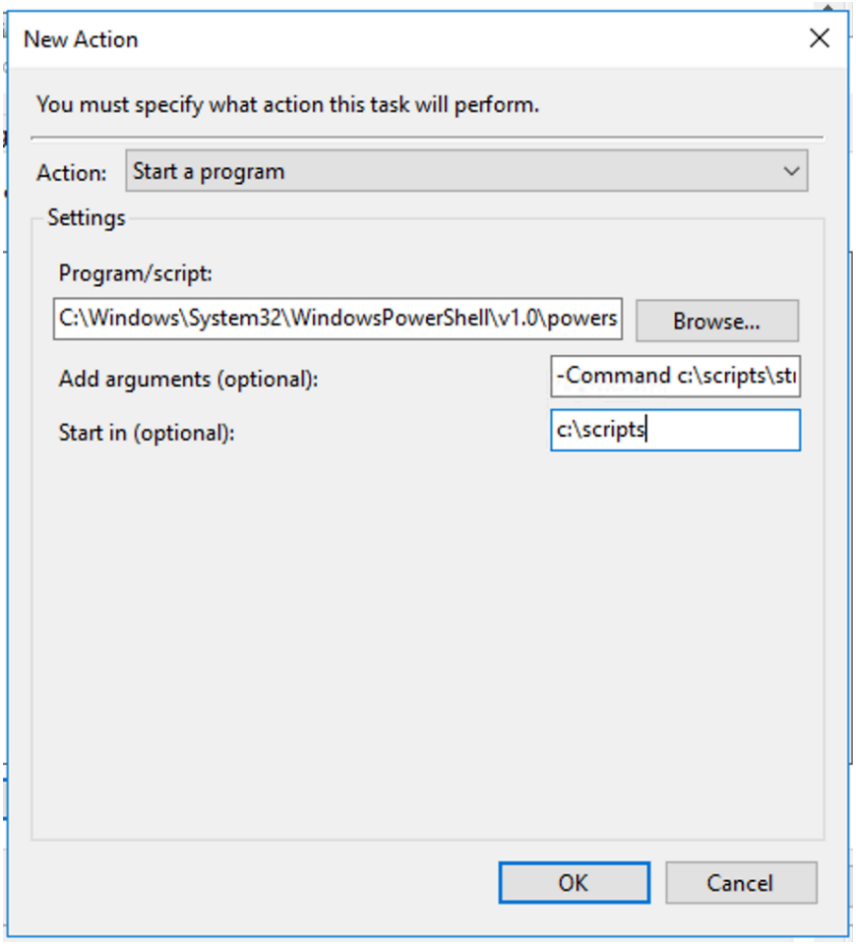
(assuming the scripts were placed in C:\Scripts)

Action:
```
Start a Program
```

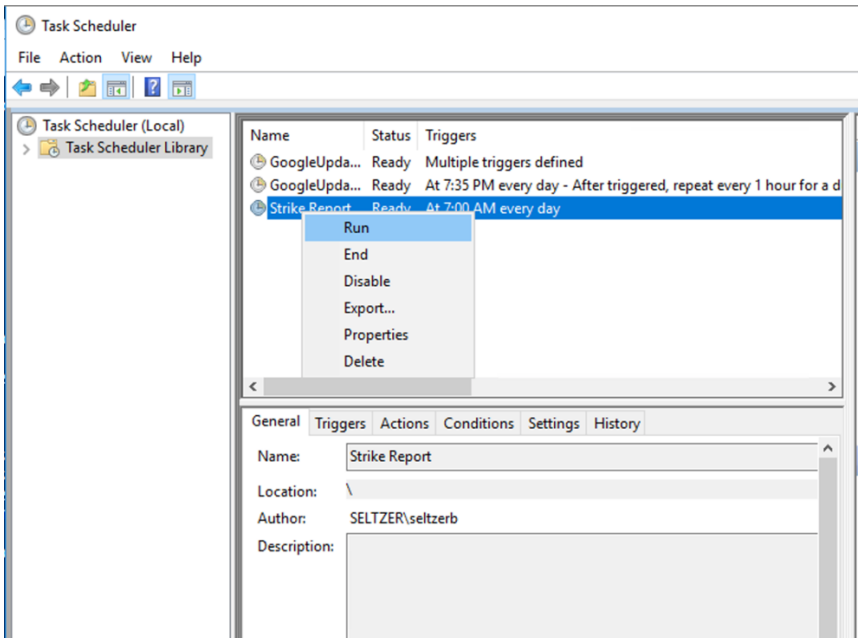
Program/script:
```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Add Arguments:
```
-command C:\Scripts\strikeReport.ps1 -vip mycluster -username myuser -domain
mydomain.net -smtpServer mail.mydomain.net -sendTo myuser@mydomain.net -
sendFrom mycluster@mydomain.net
```

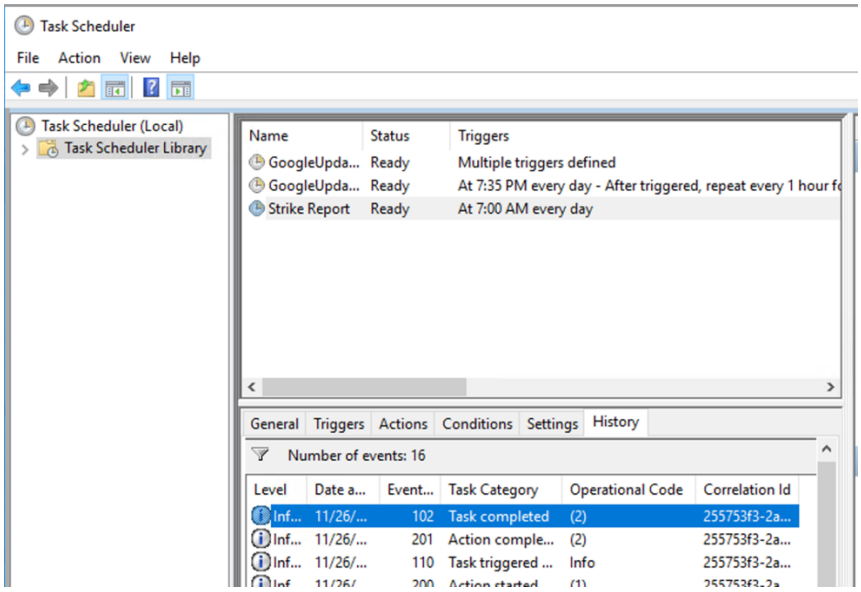
Start In:
```
C:\Scripts
```



Click OK to close the new task. You should be prompted for the password of the user who will be running the script. Now you can test-run the task. Right-click the task and click “Run”.



The status of the task will change to “Running”. Take note of the time it took when you ran the script manually. Periodically click refresh to see if the task has returned to a ready state. Also, view the history tab to see that the task completed successfully.



If the task continues to run, then the script may be prompting for input in the background, indicating that you did something wrong. Make sure the script arguments are correct and that the script runs successfully when run manually.