



**Hewlett Packard  
Enterprise**

Technical white paper

# **Building an HPE Synergy demonstration environment on your own laptop**

Includes Postman, PowerShell, Python, Ansible and  
Terraform live demonstration scenarios

## Document Revision History

Publish / Revised	Software versions	Section and Text Revised
<b>September 2020</b>	<ul style="list-style-type: none"> <li>- HPE OneView DCS <b>5.30</b> Z7550-96927</li> <li>- Windows 10 version <b>1909</b> (OS Build 18363.418)</li> <li>- Visual Studio Code version <b>1.45.1</b></li> <li>- PowerShell <b>5.1</b></li> <li>- Oracle VM VirtualBox version <b>6.1.8 r137981</b> (Qt5.6.2)</li> <li>- Windows Subsystem for Linux with Ubuntu 18.04 LTS</li> <li>- Ansible Modules for HPE OneView <b>5.7.0</b> with bug_fix/create_profiles_in_parallel</li> <li>- HPE OneView Python SDK <b>5.30</b></li> <li>- Python 2.7.17 - 3.6.9</li> <li>- Ansible <b>2.9.10</b></li> <li>- Terraform <b>0.11.14</b></li> <li>- Terraform provider for OneView <b>1.4.0</b></li> <li>- PowerShell for OneView library <b>5.2.4343.1695</b></li> </ul>	<ul style="list-style-type: none"> <li>- Update content for HPE Synergy OneView Demonstration appliance - DCS 5.30</li> <li>- Added Terraform installation in WSL and in VS Code</li> <li>- Added Terraform provider for OneView and Go installation in WSL</li> <li>- Added 2 x Terraform scenarios</li> <li>- Added Chapter-9 - How to upgrade the Synergy demonstration environment</li> <li>- Script improvements for Ansible and Python</li> <li>- Added WSL and Linux distribution recommendations to prevent incompatibility issues</li> <li>- Other minor changes</li> </ul>
<b>March 2020</b>	<ul style="list-style-type: none"> <li>- HPE OneView DCS 5.00 Z7550-96681</li> <li>- Windows 10 version 1809 (OS Build 17763.1039)</li> <li>- Visual Studio Code version 1.42.1</li> <li>- PowerShell 5.0</li> <li>- Oracle VM VirtualBox version 6.1.4 r136177 (Qt5.6.2)</li> <li>- Windows Subsystem for Linux with Ubuntu 18.04 LTS</li> <li>- Ansible Modules for HPE OneView 5.4.0 with create_profiles_in_parallel 11/27/19 fix</li> <li>- HPE OneView Python SDK 5.00</li> <li>- Python 2.7.17 - 3.6.9</li> <li>- Ansible 2.9.4</li> <li>- PowerShell for OneView library 5.0.2341.1920</li> </ul>	

<b>ABSTRACT.....</b>	<b>7</b>
<b>REQUIREMENTS .....</b>	<b>7</b>
<b>DEMONSTRATION APPLIANCE PREREQUISITES .....</b>	<b>8</b>
<b>CHAPTER-1 - DEPLOYING THE HPE ONEVIEW DEMONSTRATION APPLIANCE ON VIRTUALBOX .....</b>	<b>9</b>
DOWNLOADING HPE OneView DEMONSTRATION APPLIANCE .....	9
DOWNLOADING CENTOS 7.5 Boot ISO.....	12
DOWNLOADING AND INSTALLING VIRTUALBOX ON YOUR PC.....	12
IMPORTING AND TUNING THE HPE OneView DEMONSTRATION APPLIANCE IN VIRTUALBOX.....	15
<b>CHAPTER-2 – PREPARING UBUNTU ON WINDOWS SUBSYSTEM FOR LINUX (WSL).....</b>	<b>29</b>
INSTALLING WINDOWS SUBSYSTEM FOR LINUX (WSL).....	29
INSTALLING UBUNTU WINDOWS SUBSYSTEM FOR LINUX.....	31
INSTALLING ANSIBLE ON UBUNTU WSL .....	35
INSTALLING HPE ONEVIEW PYTHON SDK ON UBUNTU WSL .....	37
INSTALLING ANSIBLE MODULES FOR HPE ONEVIEW ON UBUNTU WSL.....	39
INSTALLING TERRAFORM ON UBUNTU WSL .....	44
INSTALLING GO ON UBUNTU WSL .....	46
INSTALLING THE TERRAFORM PROVIDER FOR HPE ONEVIEW .....	48
INSTALLING GIT FOR VS CODE SOURCE CONTROL .....	49
WHERE DO MY FILES LIVE IN MY UBUNTU WSL?.....	53
<b>CHAPTER-3 – PREPARING MICROSOFT VISUAL STUDIO CODE.....</b>	<b>54</b>
INSTALLING VISUAL STUDIO CODE.....	54
PREPARING VS CODE FOR PYTHON .....	59
PREPARING VS CODE FOR ANSIBLE .....	63
PREPARING VS CODE FOR TERRAFORM .....	64
CREATING ‘SYNERGY DEMONSTRATIONS ON WSL’ VS CODE WORKSPACE.....	65
PREPARING VS CODE FOR POWERSHELL .....	69
CREATING ‘SYNERGY DEMONSTRATIONS ON WINDOWS’ VS CODE WORKSPACE .....	73
INSTALLING THE POWERSHELL LIBRARY FOR HPE ONEVIEW.....	77
<b>CHAPTER-4 –INITIAL CONFIGURATION OF THE DEMONSTRATION APPLIANCE .....</b>	<b>79</b>
HARDWARE DISCOVERY AND INITIAL APPLIANCE CONFIGURATION .....	79
CREATION OF AN INITIAL SETUP SNAPSHOT .....	86
<b>CHAPTER-5 – PREPARING POSTMAN .....</b>	<b>89</b>
INSTALLING AND CONFIGURING POSTMAN .....	89
IMPORTING COLLECTIONS .....	90
CONFIGURATION OF THE POSTMAN ENVIRONMENT .....	93
<b>CHAPTER-6 - PREPARING YOUR DEMONSTRATION APPLIANCE FOR DEMOS.....</b>	<b>97</b>
FINAL APPLIANCE CONFIGURATION .....	98
CREATION OF A FINAL SETUP SNAPSHOT .....	107
<b>CHAPTER-7 – PREPARING THE LIVE DEMONSTRATION SCENARIOS .....</b>	<b>110</b>
DEMONSTRATING SYNERGY COMPOSER AND ONEVIEW KEY FEATURES.....	110
DEMONSTRATING INFRASTRUCTURE PROGRAMMABILITY .....	110
Postman - Scenario 1 – <i>Introduction to the OneView REST API.</i> .....	111
PowerShell – Scenario 1 - <i>Day-to-day operation task automation.</i> .....	111
PowerShell – Scenario 2 - <i>Creating a report.</i> .....	120
PowerShell – Scenario 3 - <i>Accelerating a configuration change .....</i>	123
Python – Scenario 1 - <i>Day-to-day operation task automation.</i> .....	130
Python – Scenario 2 – <i>Creating a report .....</i>	135



<i>Python – Scenario 3 - Accelerating a configuration change .....</i>	138
<i>Ansible – Scenario 1 – Collecting facts in OneView.....</i>	144
<i>Ansible – Scenario 2 – Provisioning New Servers .....</i>	154
<i>Ansible – Scenario 3 – Unprovisioning Running Servers.....</i>	164
<i>Terraform – Scenario 1 – Provisioning a new server.....</i>	168
<i>Terraform – Scenario 2 – Unprovisioning Running Server.....</i>	178
<b>CHAPTER-8 - HOW TO RESET THE SYNERGY DEMONSTRATION ENVIRONMENT.....</b>	<b>182</b>
SHUTTING DOWN THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE.....	182
RESETTING THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE TO THE FULLY CONFIGURED STATE .....	184
RESETTING THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE TO THE UNCONFIGURED STATE .....	185
<b>CHAPTER-9 - HOW TO UPGRADE THE SYNERGY DEMONSTRATION ENVIRONMENT.....</b>	<b>186</b>
UPGRADING THE HPE ONEVIEW DEMONSTRATION APPLIANCE (DCS).....	186
UPGRADING HPE ONEVIEW PYTHON SDK.....	186
UPGRADING ANSIBLE MODULES FOR HPE ONEVIEW.....	186
UPGRADING THE TERRAFORM PROVIDER FOR HPE ONEVIEW .....	186
UPGRADING THE POWERSHELL LIBRARY FOR ONEVIEW .....	187
<b>TROUBLESHOOTING VIRTUALBOX VM NOT STARTING .....</b>	<b>188</b>
<b>HARDWARE COMPONENTS USED TO BUILD THIS DEMONSTRATION ENVIRONMENT.....</b>	<b>193</b>
<b>SUMMARY.....</b>	<b>194</b>



## Abstract

If you agree that a product live demonstration is one of your best tools to strongly impact customers to adopt new technologies, then this technical white paper is for you!

Software-Defined infrastructure, infrastructure programmability (or infrastructure as code) and infrastructure automation are key features to transform, simplify and increase IT productivity. Demonstrating these unique features using a self-built demonstration environment can strongly impacts customers to adopt our technologies as they realize the value and the power of a Synergy Composable Infrastructure.

This technical white paper explains how to create a HPE Synergy demonstration (or learning) environment on your laptop (or personal computer) using the HPE Synergy Composer Demonstration appliance running on Oracle VM VirtualBox and using an Ubuntu Linux environment running on Windows 10 WSL (Windows Subsystem for Linux) to extend our demos to Python and Ansible scenarios.

This document provides all the step-by-step process for creating a simple demonstration environment by installing and configuring all required components with a HPE Synergy Composer/OneView Demonstration appliance, a lightweight Ubuntu Linux distribution fully configured with Python/Ansible/OneView modules running in the astonishing Windows Subsystem for Linux (WSL) from Microsoft, with some snapshots to reset your demonstrations and with some cool scenarios to show live demonstrations of our Infrastructure as code, total datacenter automation.

Live demonstration scenarios include:

- Postman to introduce the OneView REST API, the resource model, OneView object content, etc.
- PowerShell/Python scripts to demonstrate many aspects of the Software Defined Infrastructure, Infra as Code, etc.
- DevOps with Ansible to show how simple automation can be implemented at the Hardware Infrastructure level.

All live demonstration scripts used in this document can be found on <https://github.com/jullienl/HPE-Synergy-demonstration-environment>

For more information about HPE Synergy, please visit the HPE website. You can access to the HPE Synergy user guides and manuals at [www.hpe.com/info/synergy-docs](http://www.hpe.com/info/synergy-docs)

## Requirements

- A powerful laptop with minimum 16G of RAM (best is 32G) and with Intel Virtualization Technology enabled
- CentOS 7.5 Boot ISO
- OneView DCS 5.30 OVA File(s) for Synergy or for BL/DL (the installation and setup procedures are the same for both).
- An update version of Windows 10 (preferably April 2018 update, version 1803) to run the latest version of Windows Subsystem for Linux



## Demonstration appliance prerequisites

HPE Synergy Composer/OneView Demonstration appliance also known as the DCS (Data Center simulator) appliance has many valuable features and capabilities that contains an HPE OneView instance and a datacenter simulator with some simulated resources (Synergy frames, Synergy computes, 3PAR Storage System, etc.). Refer to the HPE OneView Demonstration Appliance Guide for more detailed information.

The DCS appliance is recommended to be deployed on a dual-core 2GHz or greater, 64-bit CPU laptop with a minimum of 16G of RAM (12GB is required for the demonstration appliance itself with 4GB allocated to the host OS and its applications).

With 16G of RAM, it is necessary to close as many applications as possible to get a smooth-running experience. For a better experience, 32G of RAM is recommended.

**Important notice:** If your laptop does not meet these requirements, you will not be able to run the appliance.

**Note:** The HPE OneView demonstration appliance is intended for demonstration purposes only and is only available to HPE employees and HPE Partners.

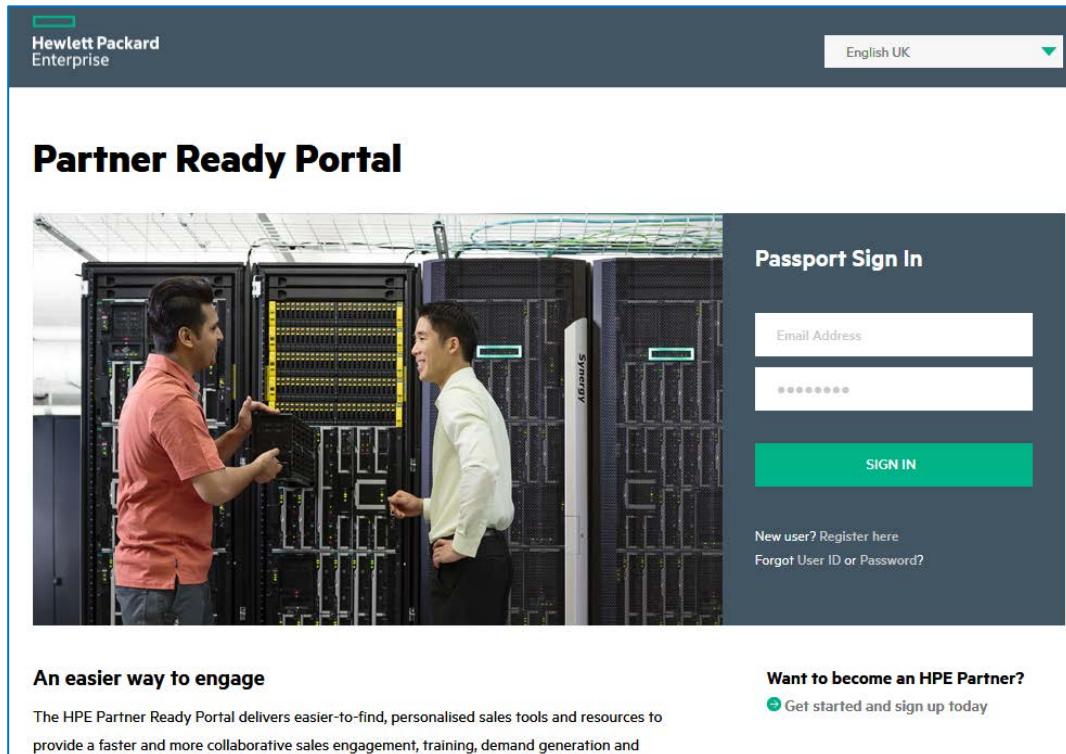


## Chapter-1 - Deploying the HPE OneView Demonstration appliance on VirtualBox

### Downloading HPE OneView demonstration appliance

Downloading location for HPE Partners:

- Log in using your HPE Passport credentials to the *HPE Partner Ready Portal*  
<https://partner.hpe.com/>

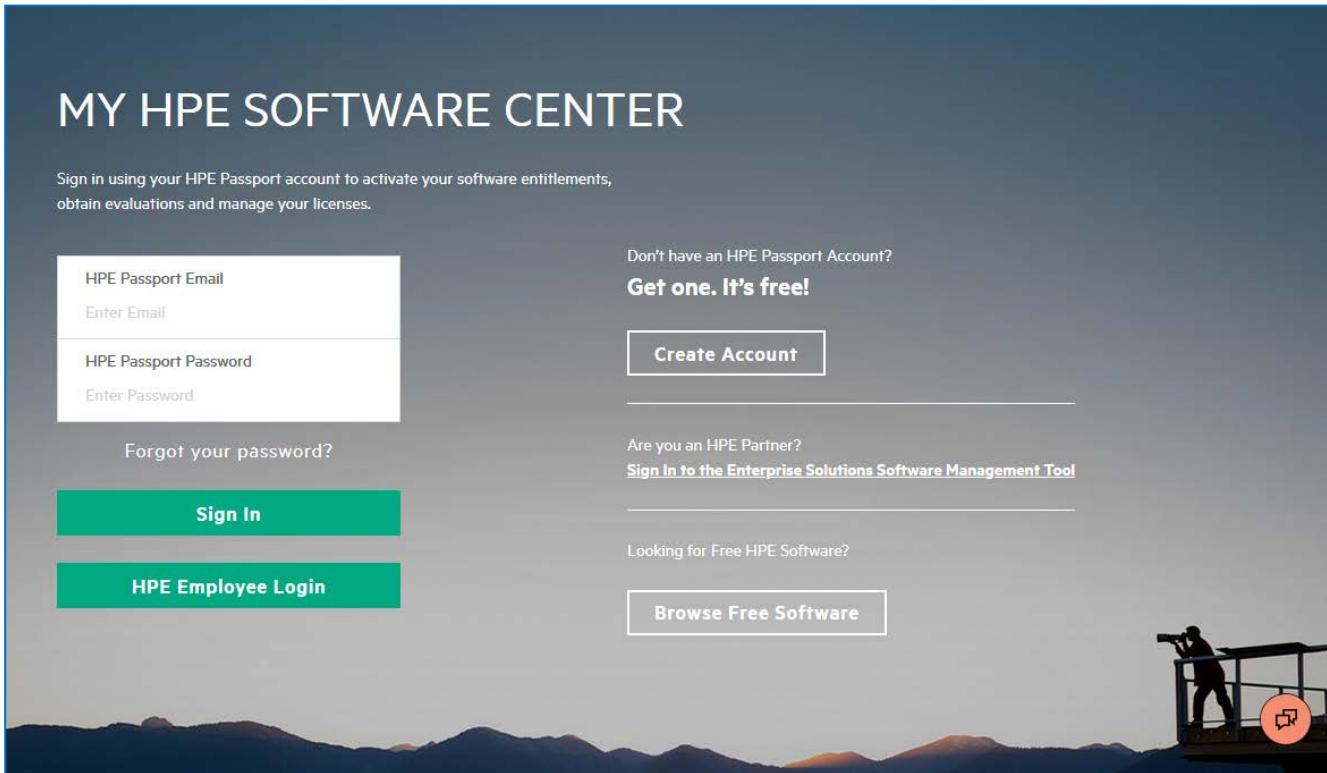


The screenshot shows the 'Partner Ready Portal' login interface. At the top left is the 'Hewlett Packard Enterprise' logo. A dropdown menu at the top right shows 'English UK'. The main title 'Partner Ready Portal' is centered above a large image of two men in a server room. To the right is the 'Passport Sign In' form with fields for 'Email Address' and 'Password', and a 'SIGN IN' button. Below the form are links for 'New user? Register here' and 'Forgot User ID or Password?'. At the bottom left, there's a section titled 'An easier way to engage' with a subtext about the portal providing personalized sales tools and resources. At the bottom right, there's a section titled 'Want to become an HPE Partner?' with a link to 'Get started and sign up today'.

- Select **My Workspace** -> **Manage Software and Licenses**
- Select **SW evaluations**
- Select **All Categories** and scroll to **OneView**
- Accept the software terms and conditions
- Select **HPE\_OneView\_DCS\_5.30\_Synergy\_ESXi\_Z7550-96927.ova** then click **Download**

Downloading location for HPE Employees:

- Visit **My HPE Software Center** portal at <https://myenterpriselicense.hpe.com/>
- Select **HPE Employee Login**



- Select **Software for HPE Employees**
- Select **OneView** in the Software Category drop down menu
- Scroll down and select **HPE OneView Demonstration Appliance**

A screenshot of the 'HPE OneView Demonstration Appliance' page. The top navigation bar includes links for Hewlett Packard Enterprise, Solutions, Services, Products, About Us, Support, and a search icon. Below the navigation is a header with a home icon, user profile, globe, and help icons. The main title 'HPE OneView Demonstration Appliance' is centered above a green 'Download' button. A descriptive text box below the title states: 'The HPE OneView demonstration appliance provides the latest version of HPE OneView software along with a simulation of HPE BladeSystem c-class/DL servers or HPE Synergy infrastructure.' To the right of this text is a 'Leave Feedback' button.

**Important notice:** HPE OneView demonstration appliance is for internal and channel partner use only. It should not be given to customers.

- Accept the software terms and conditions
- Select **HPE\_OneView\_DCS\_5.30\_Synergy\_ESXi\_Z7550-96927.ova** then click **Download**

Download Files

Software (30)

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_SYNERGY\\_ESXI\\_Z7550-96927.OVA](#) (2.24 GB)  
SHA256 Checksum: 624ba77ef5fdeb0af04e19b9c9557c03f518864c684a4af2bf... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_SYNERGY\\_HYPER\\_V\\_Z7550-96928.ZIP](#) (1.95 GB)  
SHA256 Checksum: 2bdd689a6e7631e24569872d245bad1cd10421ef3e7b5fe6d0... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_SYNERGY\\_KVM\\_Z7550-96929.TAR.GZ](#) (2.18 GB)  
SHA256 Checksum: b2d114dbc799d7b76ffff5b681bc27071b377443f911ec8b88... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_VIRTUAL\\_ESXI\\_Z7550-96930.OVA](#) (2.24 GB)  
SHA256 Checksum: 4e20f9a92c105d52c18a0ac22e91bc13c5405df3162a6bd210... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_VIRTUAL\\_HYPER\\_V\\_Z7550-96931.ZIP](#) (1.95 GB)  
SHA256 Checksum: 4c904d19d2aba336e5628969e191d68c7f2465b04f24130d5e... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.30\\_VIRTUAL\\_KVM\\_Z7550-96941.TAR.GZ](#) (2.18 GB)  
SHA256 Checksum: 9d198fbbeaac2cdf3d04ba457d4ded2485dcfae67020d1f47... ([Copy](#))

[HPE\\_ONEVIEW\\_DCS\\_5.20\\_SYNERGY\\_ESXI\\_Z7550-96879.OVA](#) (2.22 GB)  
SHA256 Checksum: f8e32b3733c26719eff9935b46ad96b1b28d78a37895034821... ([Copy](#))

Tell me what I can do here... ▾

**Download Files**  
The files displayed in this screen depend on the product(s) activated including: licenses/keys, software and/or documentation.  
There are 3 alternatives to download files:

- Download all files - Check the box at the top of each section and click the Download button.
- Download specific files - Check the box next to each file and click the Download button.
- Download individual file - Click on the hyperlink of the file.

For the license keys you can also copy the license key directly to the clipboard by clicking the copy link next to the key.  
To [Email Licenses/Keys](#) to another user or [View Activation Details](#), click the respective link at the top of this section.

**Terms & Conditions**  
By activating the licenses and/or downloading the software from this site, you accept these [Terms and Conditions](#).

Feedback icon

## Downloading CentOS 7.5 Boot ISO

CentOS 7.5 ISO is required to perform a “Rescue” on the HPE OneView demonstration appliance VM in order to change two items to make the appliance boot successful on VirtualBox.

- Download the ISO from [http://mirrors.usc.edu/pub/linux/distributions/centos/7.5.1804/isos/x86\\_64/CentOS-7-x86\\_64-Minimal-1804.iso](http://mirrors.usc.edu/pub/linux/distributions/centos/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso)

**Note:** The above file is approximately 1GB and is no longer the current version of CentOS so only some mirrors will have it.

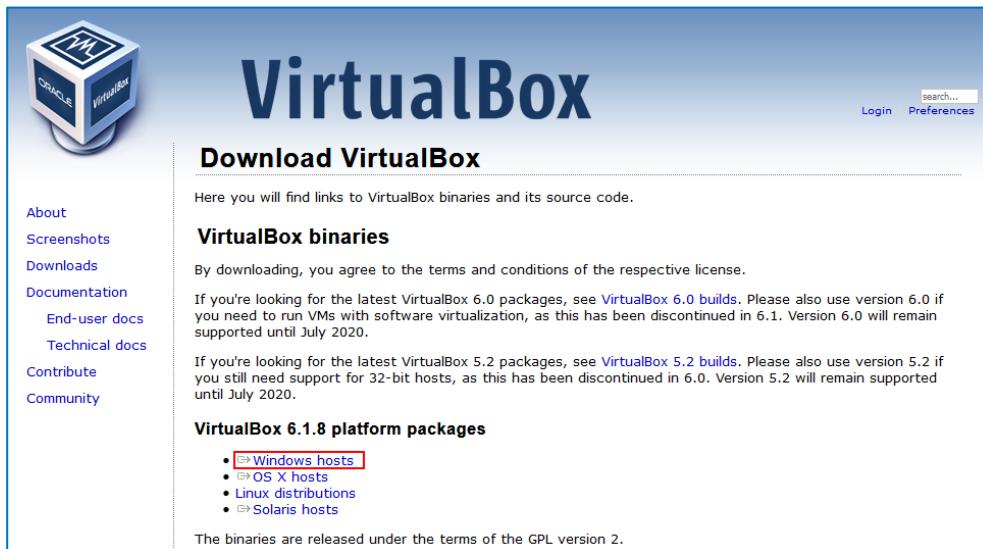
## Downloading and installing VirtualBox on your PC

Oracle VM VirtualBox is a free and open-source hosted hypervisor for x86 virtualization, developed by Oracle Corporation. VirtualBox offers many of the VMware Workstation features, and couple of unique ones.

VMware Workstation is free during the trial evaluation period but after that, you'll need to buy a license. This is the reason why VirtualBox from Oracle is becoming a good alternative to run the HPE Demonstration Appliance for HPE Synergy (aka DCS).

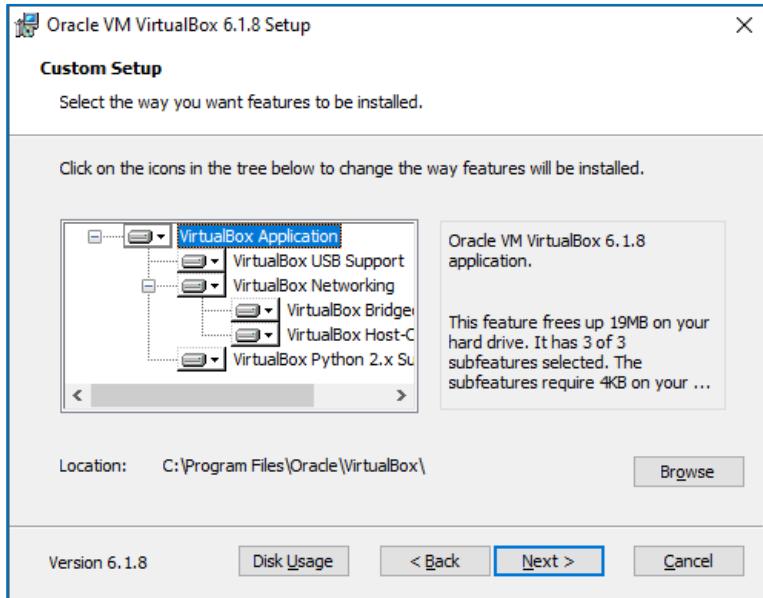
As mentioned in the prerequisites, the HPE OneView demonstration appliance is not officially supported with a VirtualBox hypervisor. The main reason for that is simply because it has not been tested and validated by HPE, but the experience shows that everything is working properly and as expected if you follow the right procedure.

- To download VirtualBox, go to <https://www.virtualbox.org/wiki/Downloads>
- Select **Windows hosts**

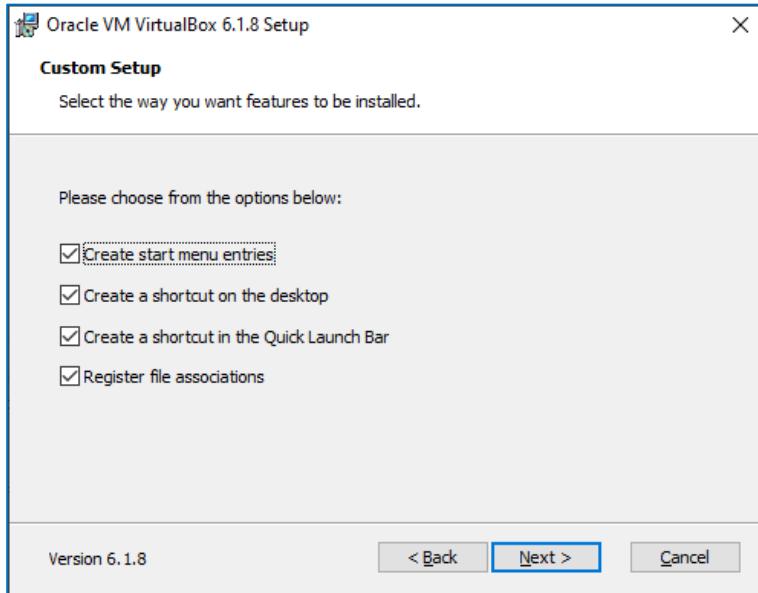


- Save the file on your PC and launch the executable.

- Leave all default parameters then click **Next**



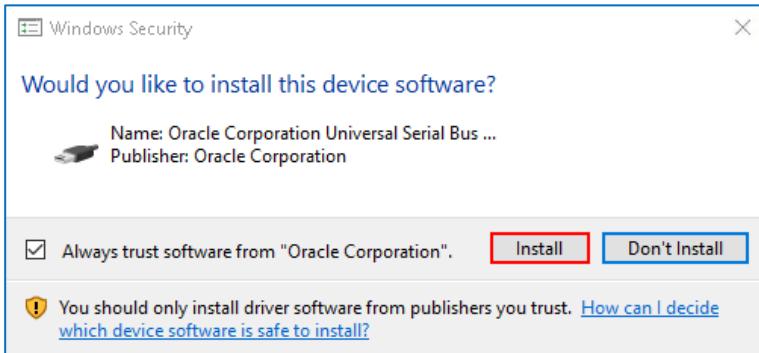
- Click **Next**



- Click **Next**



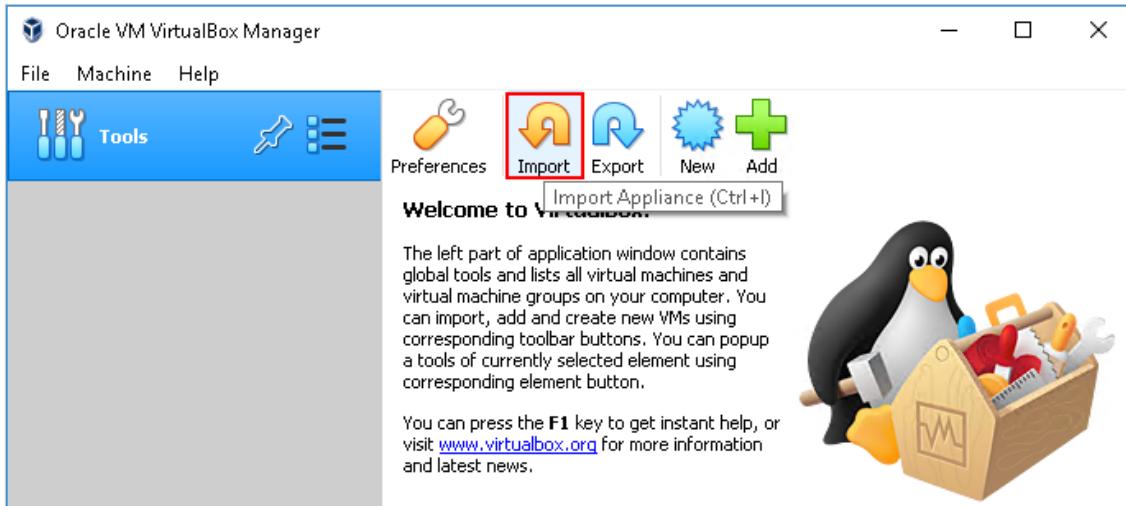
- Then click **Yes** to proceed with the installation then click **Install**
- When the Windows Security message pops-up, click **Install**



- Then click **Finish** with the start option checked

## Importing and tuning the HPE OneView Demonstration appliance in VirtualBox

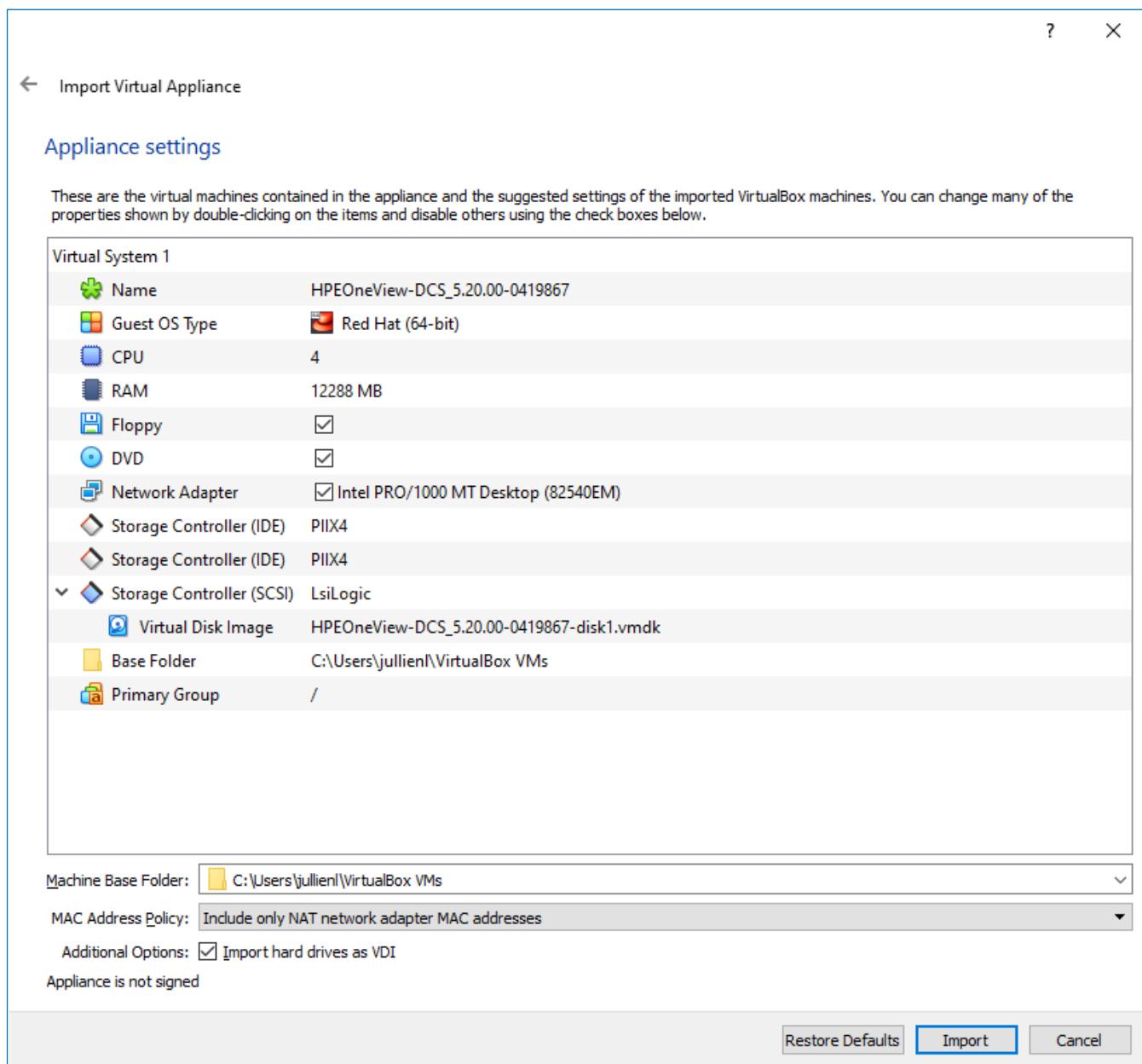
- In VirtualBox, select **Import**



- Select the **HPE\_OneView\_DCS\_x.xx\_Synergy\_ESXi\_Zxxxx.ova** file



- Leave all default parameters then click **Import**



**Note:** If you start the VM in the current state, the VM is entering in emergency mode because the driver of the disk controller simulated by VirtualBox is not available.

```
HPEOneView-DCS_5.20.00-0419867 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
exist
[ 212.812407] dracut-initqueue[290]: Warning: /dev/vg01/lv_root does not exist
[ 212.813896] dracut-initqueue[290]: Warning: /dev/vg01/lv_swap does not exist
    Starting Setup Virtual Console...
[ OK ] Started Setup Virtual Console.
[ 213.999421] type=1130 audit(1590600065.519:11): pid=1 uid=0 auid=4294967295 ses=4294967295 subj=kernel msg='unit=systemd-vconsole-setup comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
    Starting Dracut Emergency Shell...
Warning: /dev/mapper/vg01-lv_root does not exist
Warning: /dev/vg01/lv_root does not exist
Warning: /dev/vg01/lv_swap does not exist

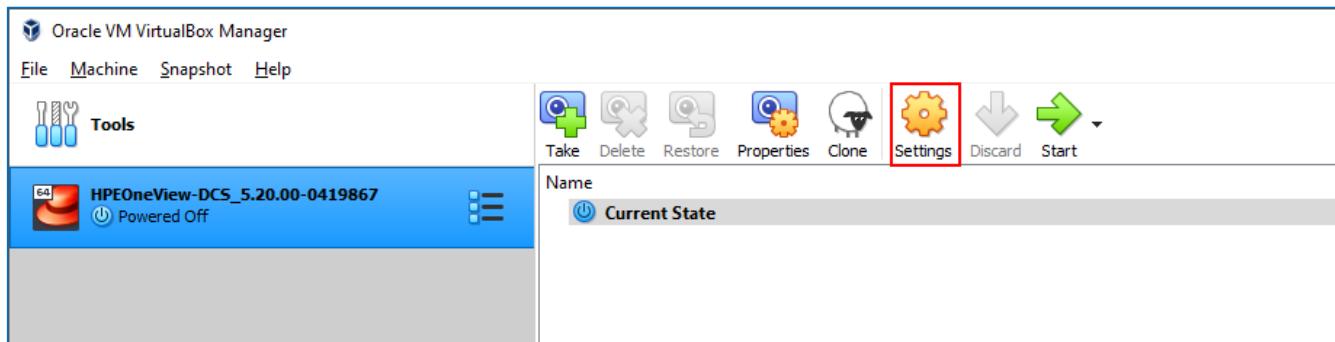
Generating "/run/initramfs/rdsosreport.txt"
[ 214.856670] blk_update_request: I/O error, dev fd0, sector 0
[ 214.879431] blk_update_request: I/O error, dev fd0, sector 0

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot after mounting them and attach it to a bug report.

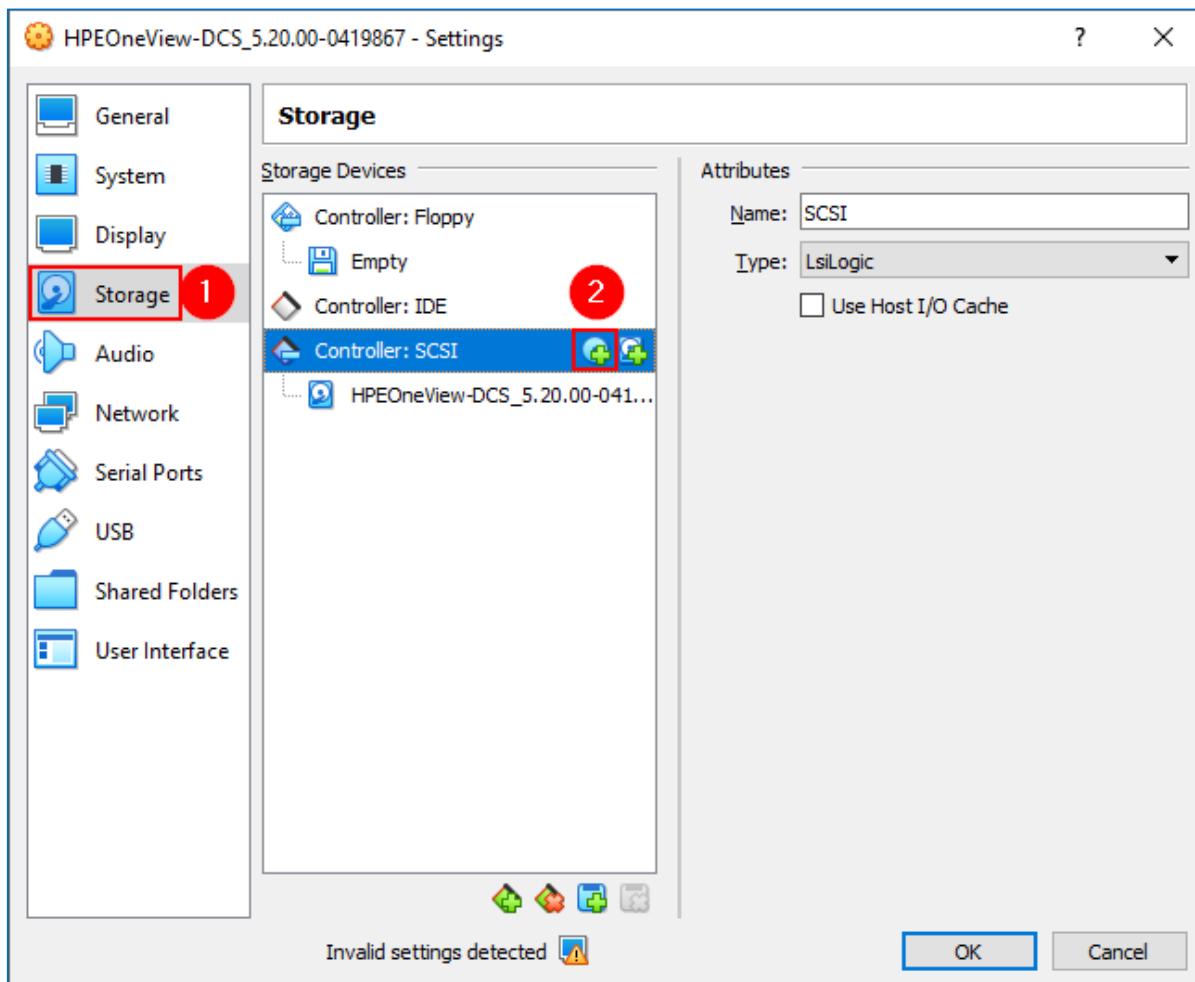
dracut:/# _
```

The solution to fix this issue is to inject the missing drivers from the downloaded CentOS 7.5 CD ISO.

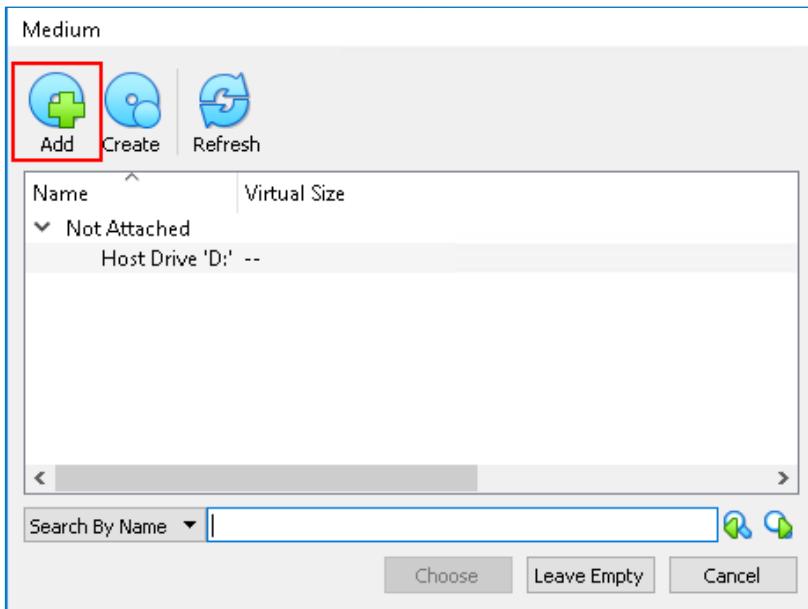
- Edit the VM Settings



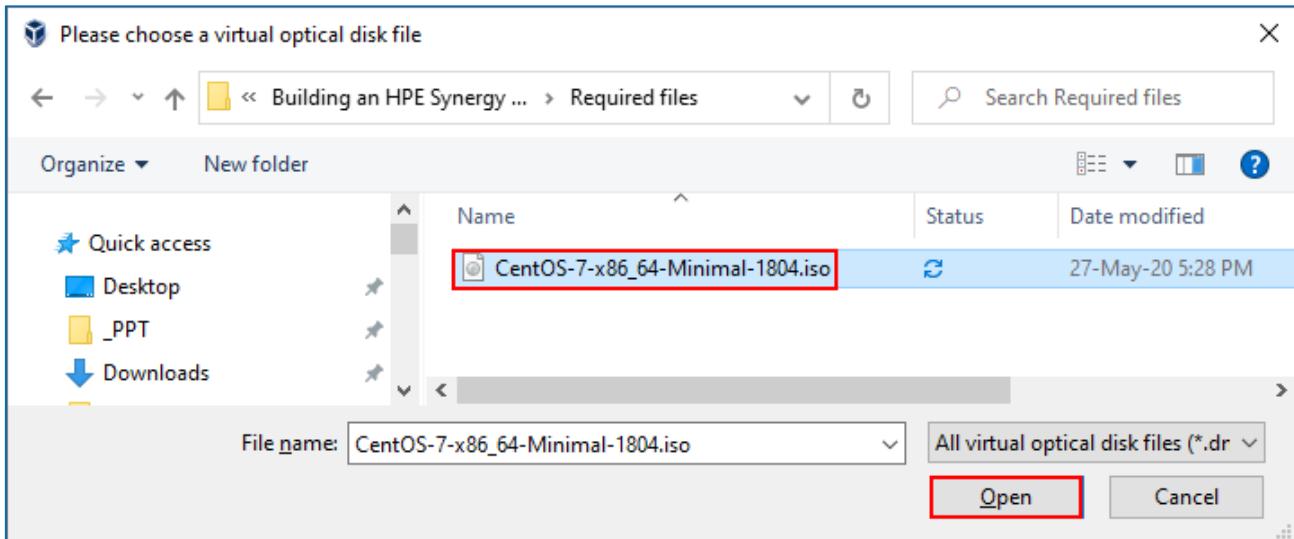
- Select **Storage** then click on **Adds optical drive** icon on the iSCSI Controller



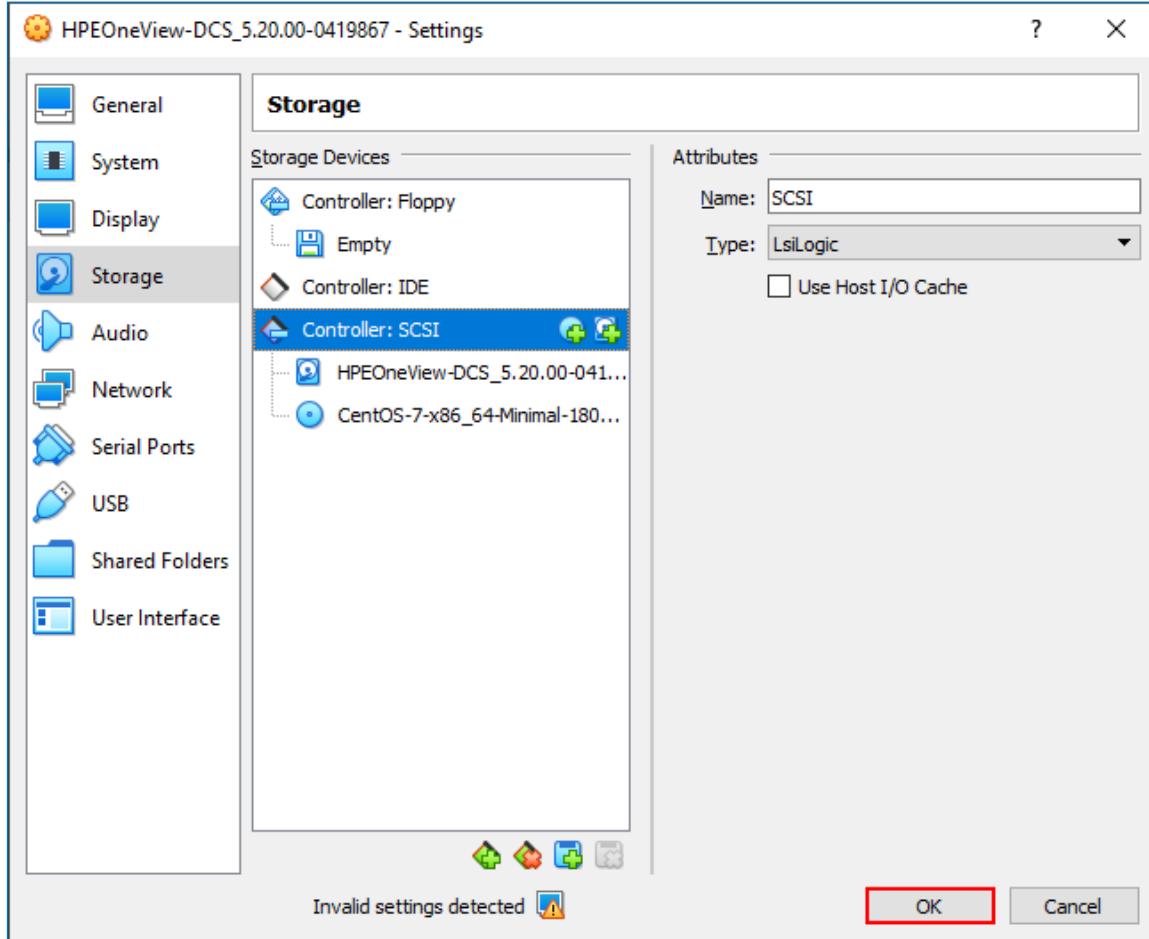
- Select **Add**



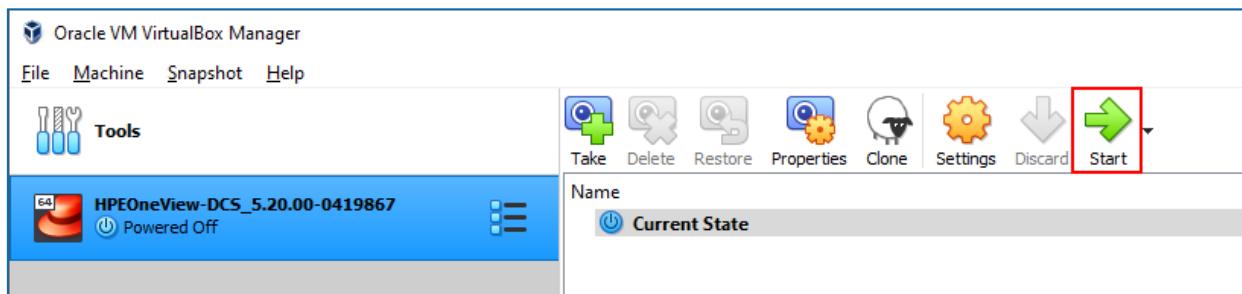
- Select the CentOS ISO image then click **Open**



- Click on **Choose** then **OK**



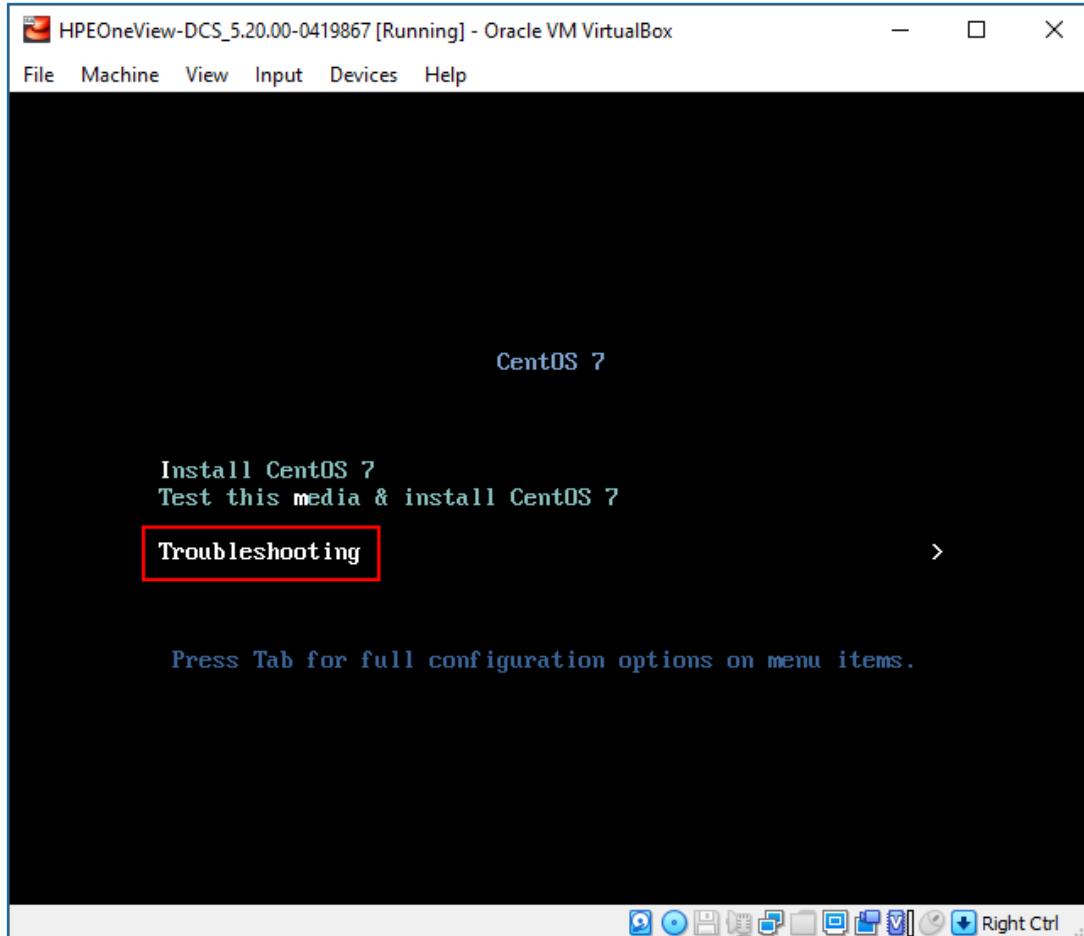
- Then start the VM by pressing **Start**



The CentOS starting menu should come up.

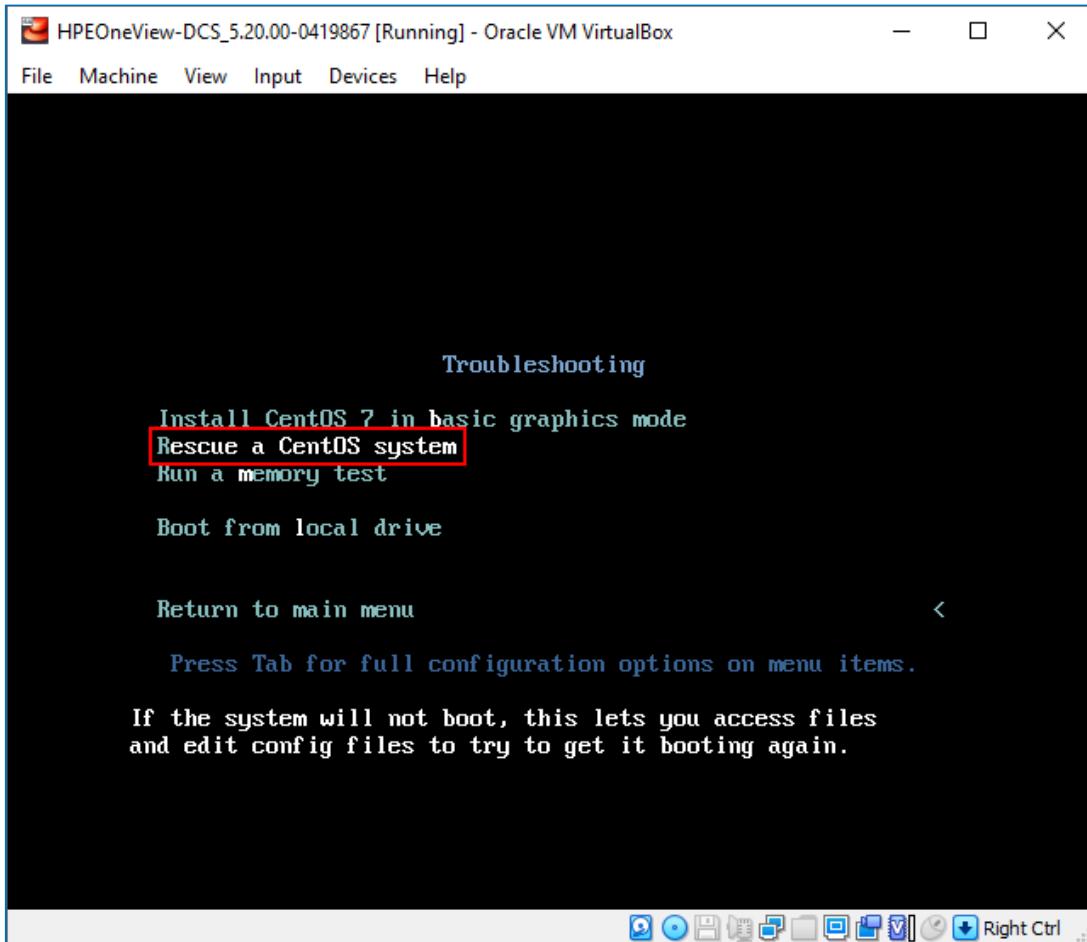
**Note:** If you are seeing some errors when you start the VM, go to the [Troubleshooting](#) section

- Select **Troubleshooting**

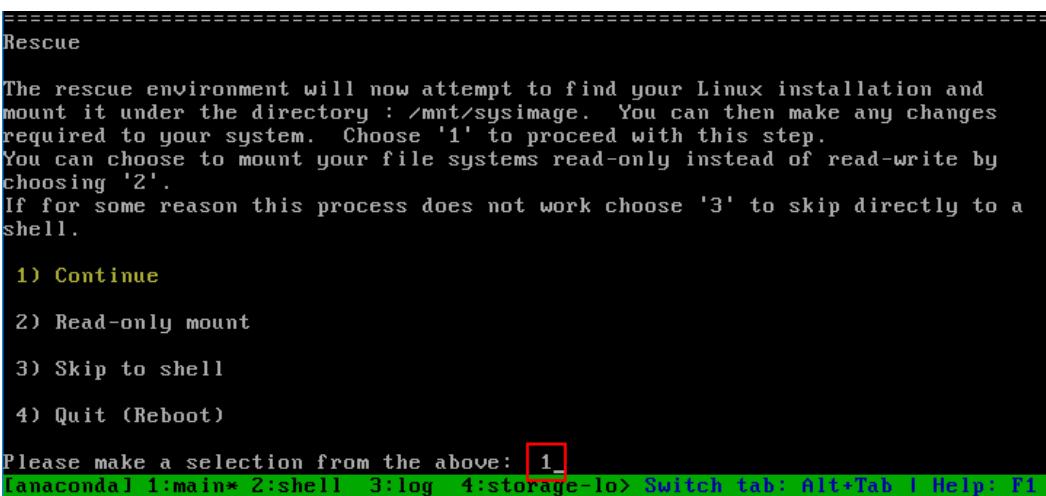


**Note:** To release the mouse from a VirtualBox console use the right Control key on your keyboard

- Then select **Rescue a CentOS system**



- Select option **1** to continue with Rescue Mode.



- Press **ENTER** to get a shell.

```
Rescue Mount  
Your system has been mounted under /mnt/sysimage.  
If you would like to make your system the root environment, run the command:  
    chroot /mnt/sysimage  
Please press <return> to get a shell.  
[anaconda] 1:main* 2:shell 3:log 4:storage-lo> Switch tab: Alt+Tab | Help: F1
```

- Run the following command mentioned in the console to mount the disk drive from the appliance:

```
chroot /mnt/sysimage
```

**Note:** You can use **TAB** to Autocomplete commands

```
=====  
Rescue Mount  
Your system has been mounted under /mnt/sysimage.  
If you would like to make your system the root environment, run the command:  
    chroot /mnt/sysimage  
Please press <return> to get a shell.  
When finished, please exit from the shell and your system will reboot.  
sh-4.2# chroot /mnt/sysimage  
bash-4.2#  
[anaconda] 1:main* 2:shell 3:log 4:storage-lo> Switch tab: Alt+Tab | Help: F1
```

Your command prompt should change from *sh* to *bash*

- Run the following to replace the Boot Image with an alternate that contains the drivers we need.

```
cd /boot  
cp initramfs-0-rescue-8dbf57addb634b6c8db1e628a7275adb.img initramfs-3.10.0-862.3.2.el7.x86_64.img
```

**Note:** The filenames may be slightly different between BL/DL and Synergy schematics, use TAB auto complete and you should only need to type the parts in bold followed by TAB in order to get the proper command syntax.

- After running this command, you will receive no feedback, just a prompt.

```
bash-4.2# cp initramfs-0-rescue-8dbf57addb634b6c8db1e628a7275adb.img initramfs-3  
.10.0-862.3.2.el7.x86_64.img  
bash-4.2#  
[anaconda] 1:main* 2:shell 3:log 4:storage-lo> Switch tab: Alt+Tab | Help: F1
```

As part of the OneView Security Hardening process, SELinux has been enabled. On some laptops while testing, we observed that SELinux gets in the way and the DCS does not load properly. Perform the following steps to relax the SELinux settings.

- Change to the SELinux Config folder:

```
cd /etc/selinux
```

- Open the config file for editing:

```
vi config
```

- Press the letter **i** to put the VI editor in *Insert Mode*. Use the arrow keys to move down to the **SELINUX= entry** and change it from **enforcing** to **permissive**

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- Press **ESC** to exit *Insert Mode*, then type **:** (colon) to open the vi command prompt, type **wq** and press **ENTER** to write and quit vi
- Type the exit command to unmount the appliance Image and return to the Rescue Mode shell.:

```
exit
```

```
bash-4.2# vi config
bash-4.2# exit
exit
sh-4.2#
[anaconda] 1:main* 2:shell 3:log 4:storage-> Switch tab: Alt+Tab | Help: F1
```

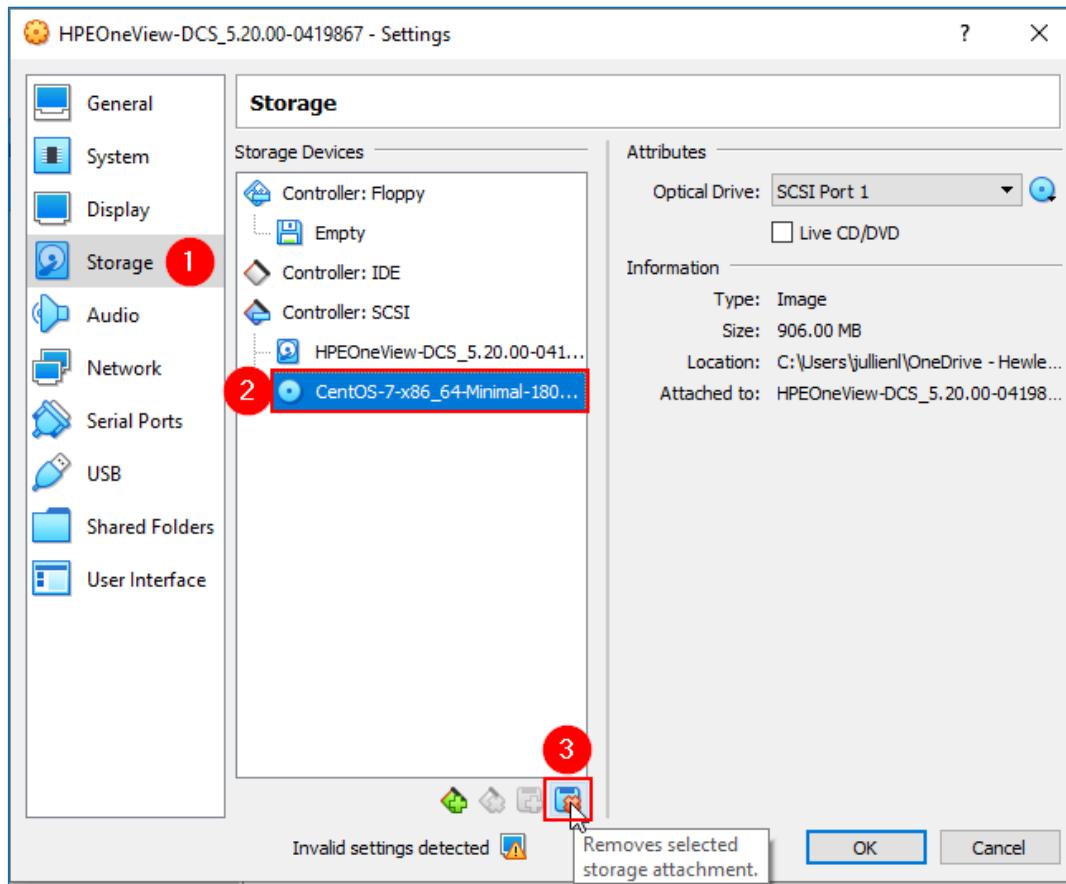
The command prompt should change from *bash* back to *sh*.

- Then shut down the VM:

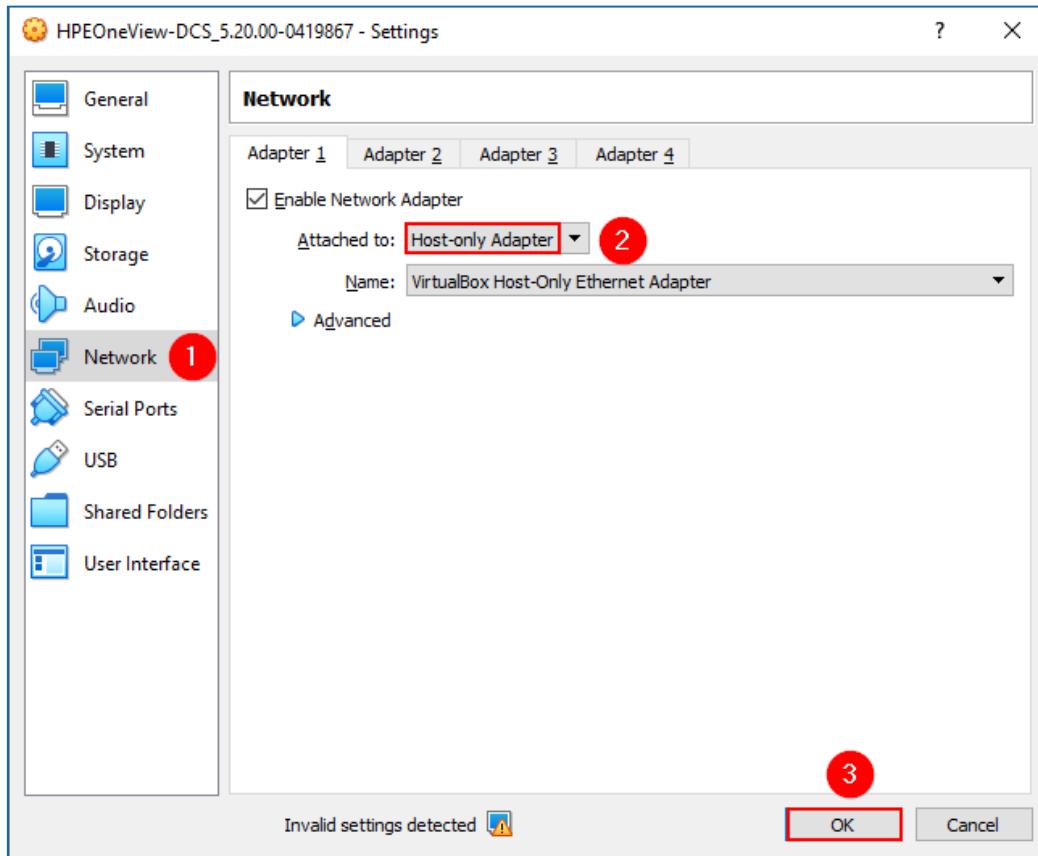
```
shutdown now
```

```
[ OK ] Unmounted /mnt/sysimage/dev/shm.
[ OK ] Unmounted /mnt/sysimage/dev/pts.
[ OK ] Unmounted /mnt/sysimage/boot.
  Unmounting /mnt/sysimage/dev...
  Unmounting /mnt/sysimage/sys...
[ OK ] Unmounted Temporary Directory.
[ OK ] Unmounted /mnt/sysimage/sys.
[ OK ] Unmounted /mnt/sysimage/dev.
  Unmounting /mnt/sysimage...
[ OK ] Stopped target Swap.
  Deactivating swap /dev/dm-9...
[ OK ] Deactivated swap /dev/vg01/lv_swap.
[ OK ] Deactivated swap /dev/mapper/vg01-lv_swap.
[ OK ] Deactivated swap /dev/disk/by-uuid/...030-c193-4454-a09a-43ca1b824832.
[ OK ] Deactivated swap /dev/disk/by-id/dm...LGNt6qt19MViJaJSQ9Mm0de0TjV8hxJi.
[ OK ] Deactivated swap /dev/disk/by-id/dm-name-vg01-lv_swap.
[ OK ] Deactivated swap /dev/dm-9.
[ OK ] Unmounted /mnt/sysimage.
[ OK ] Reached target Unmount All Filesystems.
[ OK ] Stopped target Local File Systems (Pre).
[ OK ] Stopped Remount Root and Kernel File Systems.
  Stopping Remount Root and Kernel File Systems...
[ OK ] Stopped Create Static Device Nodes in /dev.
  Stopping Create Static Device Nodes in /dev...
```

- Once the VM has shutdown, edit the VM **Settings** and remove the CentOS CD.



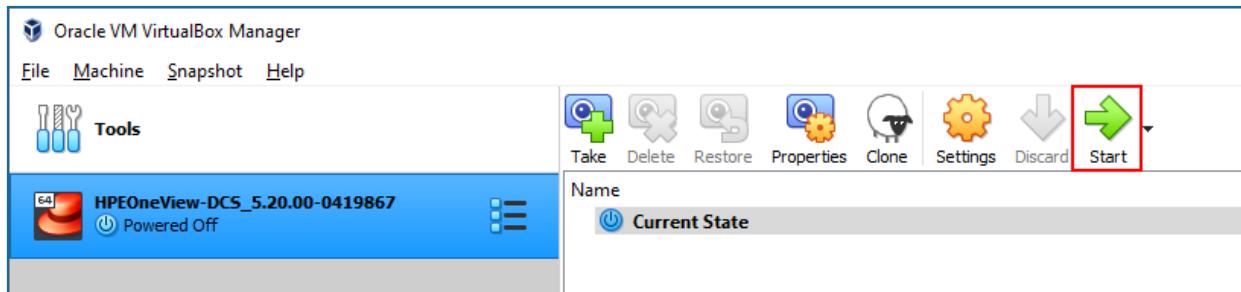
- Then we need to configure the VM network to use **Host-only Adapter** because the appliance requires the network NOT to use DHCP:



- Click **OK** to save changes.
- Now it is very important to shut down all applications running on your computer before initiating the startup process. This relieves memory/CPU/disk pressure on the VM while booting.

**Important notice:** It is recommended that you close all programs before starting the DCS appliance as the lack of CPU/memory resources during startup and hardware discovery can negatively impact the appliance. Once the appliance is started and the initial discovery is complete, the lack of resources is not much of a concern.

- Click on **Start** to power on the HPE OneView demonstration appliance VM.

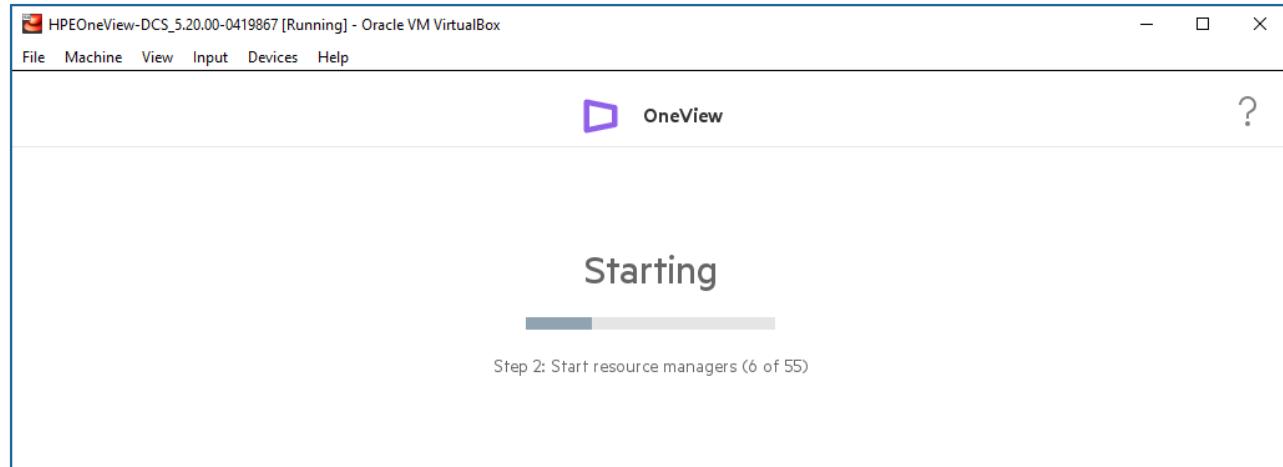


- At this point it should begin its normal unpacking process which may reboot the VM a few times.

```
[ OK ] Started Rebuild Hardware Database.
      Starting udev Coldplug all Devices...
[ OK ] Started udev Coldplug all Devices.
      Starting udev Wait for Complete Device Initialization...
[ OK ] Found device /dev/mapper/vg01-lv_swap.
      Activating swap /dev/mapper/vg01-lv_swap...
[ OK ] Activated swap /dev/mapper/vg01-lv_swap.
[ OK ] Reached target Swap.
[ OK ] Created slice system-lvm2\|x2dpvscan.slice.
      Starting LVM2 PV scan on device 8:2...
[ OK ] Found device HARDDISK 1.
[ OK ] Started LVM2 PV scan on device 8:2.
[ OK ] Reached target Sound Card.
[ OK ] Started udev Wait for Complete Device Initialization.
      Starting Activation of LVM2 logical volumes...
[ OK ] Found device /dev/mapper/vg01-lv_backup_staging.
[ OK ] Found device /dev/mapper/vg01-lv_files_rep.
[ OK ] Found device /dev/mapper/vg01-lv_db_rep.
[ OK ] Found device /dev/mapper/vg01-updatelogs.
[ OK ] Found device /dev/mapper/vg01-lv_tmp.
[ OK ] Started Activation of LVM2 logical volumes.
[ OK ] Found device /dev/mapper/vg01-lv_var.
[ OK ] Reached target Local Encrypted Volumes.
      Starting Activation of LVM2 logical volumes...
```



- When it changes from the Text mode to a GUI, it should be between 15-35 minutes (depends on HDD/SSD speed) before the DCS appliance is fully booted and ready for use.



**Note:** If your laptop is running with 16GB of memory, it is recommended to shut down as many applications as possible to ensure best performance when running the DCS appliance. Consider also closing temporarily background programs like Skype, OneDrive, etc.

**Note:** If during the boot a Maintenance password is requested, just press **CTRL + D** to continue.

At this stage you do not need to wait for the boot to complete, continue to the next chapter.

This concludes Chapter-1

In the next chapter, we will install and configure Windows Subsystem for Linux.



## Chapter-2 – Preparing Ubuntu on Windows Subsystem for Linux (WSL)

### Installing Windows Subsystem for Linux (WSL)

Windows Subsystem for Linux (WSL) is an optional feature on Windows 10 that creates a lightweight environment that allows you to install and run supported versions of Linux (such as Ubuntu, OpenSUSE, Debian, etc.) without the complexity and overhead of a virtual machine. It is light, fast and easy to use.

To learn more, see <https://docs.microsoft.com/en-us/windows/wsl>

For our PC demonstration environment, we are going to use WSL (version 1) to run Python, Go, Ansible and Terraform.

**Note:** It is recommended to use a recent version of WSL (Windows 10, April 2018 update, version 1803) to get better performance, more compatibility with Linux-native applications and VS Code extensions. (You can run `winver` to find the Windows version you are running).

**Note:** The old version of WSL in Windows version 1709 can also be used for this lab.

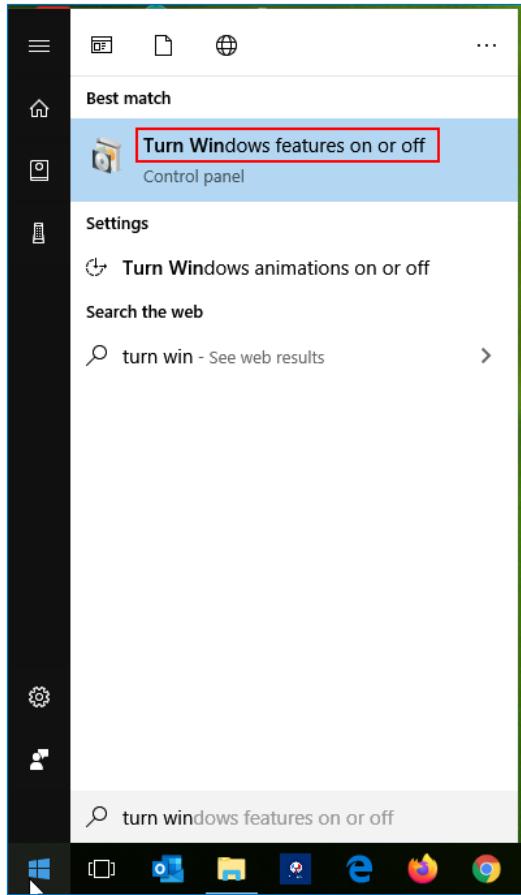
**Note:** WSL2 is a new version of the architecture in WSL that is increasing file system performance and adding full system call compatibility, see <https://docs.microsoft.com/en-us/windows/wsl/wsl2-index>.

WSL 2 is only available in Windows 10, Version 2004, Build 19041 or higher.

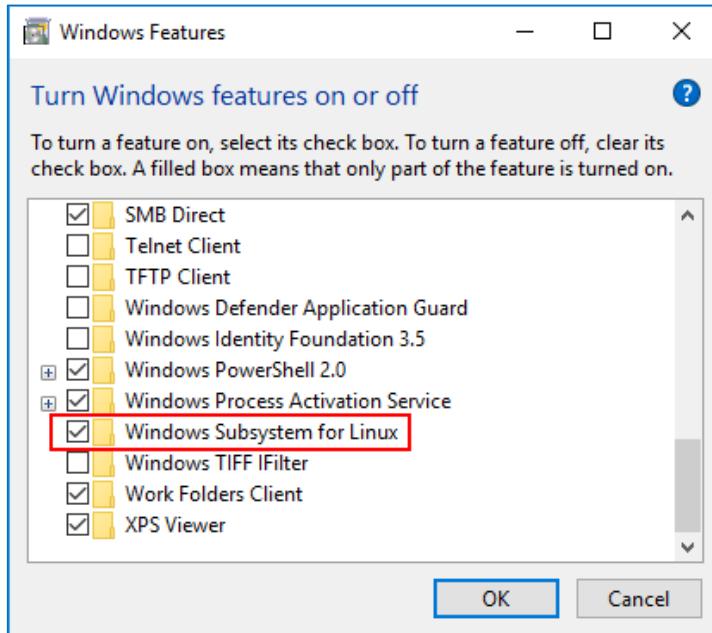
We do not recommend using WSL2 as there are many known compatibility issues with Oracle VM VirtualBox as unlike WSL, WSL2 is using Hyper-V and for now, Hyper-V does not play nicely with any other virtualization platform.

- To enable the WSL feature, open the Windows start menu and enter **Turn windows**
- Then select **Turn Windows Features on and off**





- Select the **Windows Subsystem for Linux** check box.



- Once turned on, follow the instructions, and do any restarts if you have to.



## Installing Ubuntu Windows Subsystem for Linux

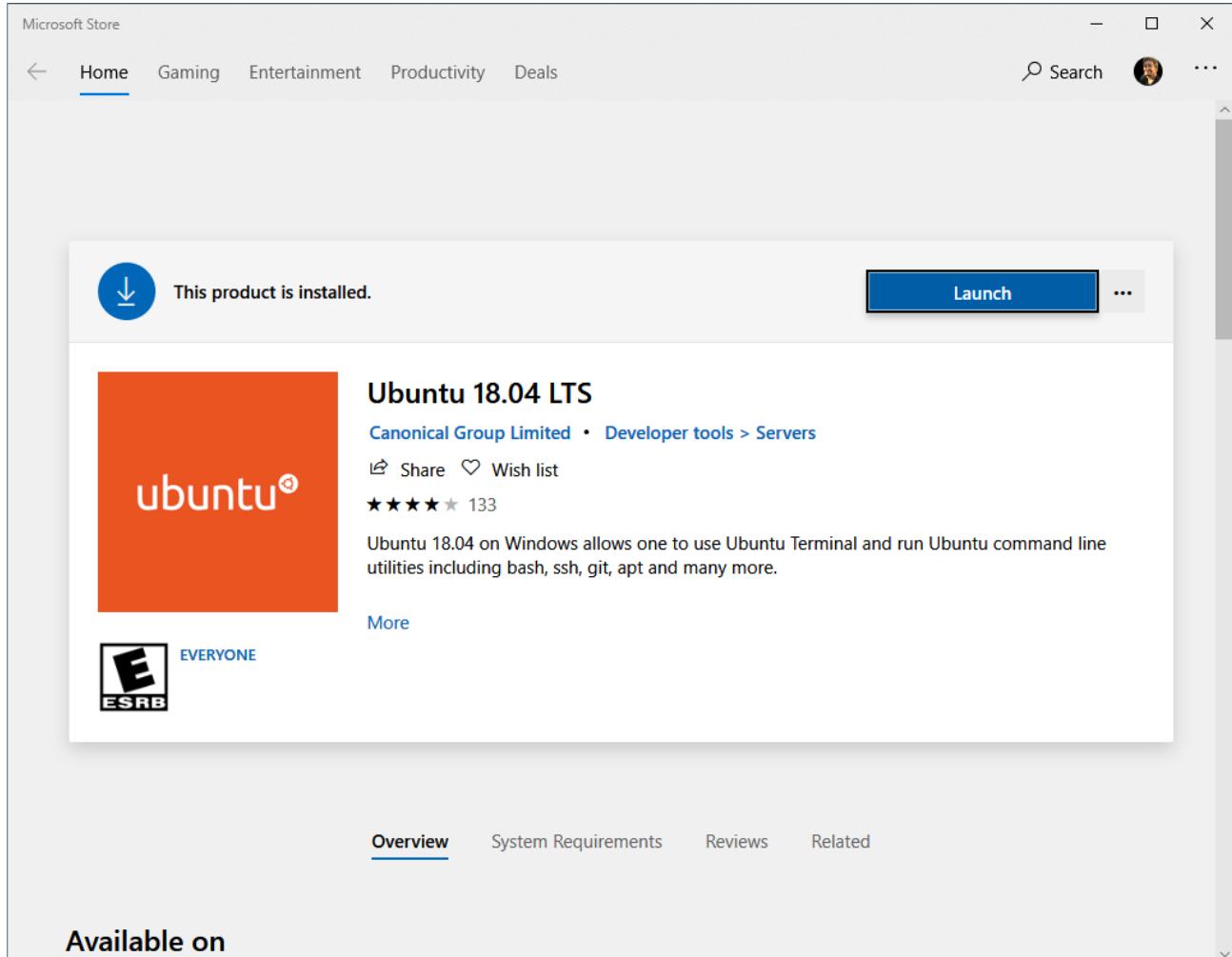
Next, we need to install a Linux distribution, we are going to use Ubuntu since that is one of the most popular.

**Important notice:** There are some issues running Ubuntu 20.04 LTS on WSL (version 1) so we highly recommend Ubuntu version 18.04. For more information, see <https://discourse.ubuntu.com/t/ubuntu-20-04-and-wsl-1/15291>

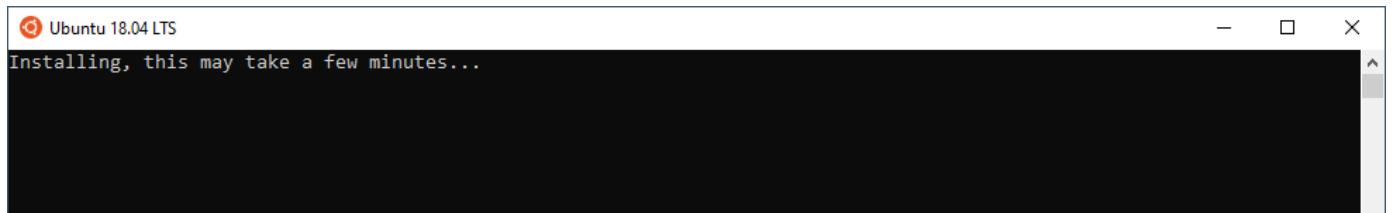
- To install Ubuntu 18.04 LTS, open the Microsoft Store at <https://www.microsoft.com/en-us/p/ubuntu-1804-lts/9n9tngvndl3q>
- Then select **Get**

The screenshot shows the Microsoft Store page for the "Ubuntu 18.04 LTS" app. At the top, there's a navigation bar with links like Home, Devices, Software, Games & Entertainment, Deals, Shop Business, More, and a sign-in link. Below the navigation is a search bar and a cart icon. The main content area features a large orange tile with the "ubuntu" logo. To the right of the tile, the text "Ubuntu 18.04 LTS" is displayed, along with the developer information "Canonical Group Limited" and the category "Developer tools > Servers". A "Free" badge is prominently shown, followed by a "Get" button and a "See System Requirements" link. Below the main tile, there's a section titled "Available on" showing a "PC" icon. Under "Description", there's a brief overview of what the app does, a note about system requirements, and instructions for launching it from the Start Menu. At the bottom of the page, there are tabs for "Overview", "System Requirements", "Reviews", and "Related".

- Once Ubuntu has been downloaded and installed, click the **Launch** button in the Microsoft Store app, or launch Ubuntu from the Start menu or type `wsl` in a windows command prompt.

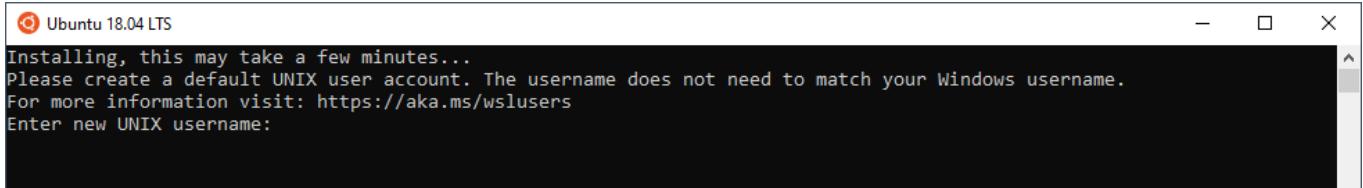


- The first time a newly installed distro runs, a Console window will open, and we will be asked to wait for a minute or two for the installation to complete.



Note that the installation may take longer time depending on the performance of your PC's storage devices. This initial installation phase is only required when a distro is clean-installed - all future launches should take less than a second.

- Once installation is complete, you will be prompted to create a new user account (and its password).

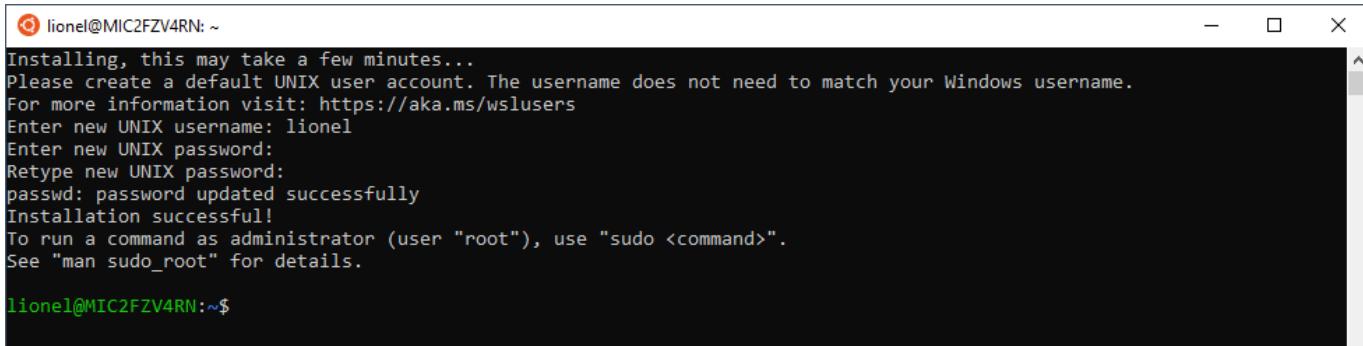


```
Ubuntu 18.04 LTS
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

- This user account is for the normal non-admin user that you will be logged-in as by default when launching Ubuntu.

**Note:** You can choose any *username* and *password* you wish - they have no bearing on your Windows username.

**Note:** When you open a new distro instance, you won't be prompted for your password, but if you elevate a process using *sudo*, you will need to enter your password, so make sure you choose a password you can easily remember!

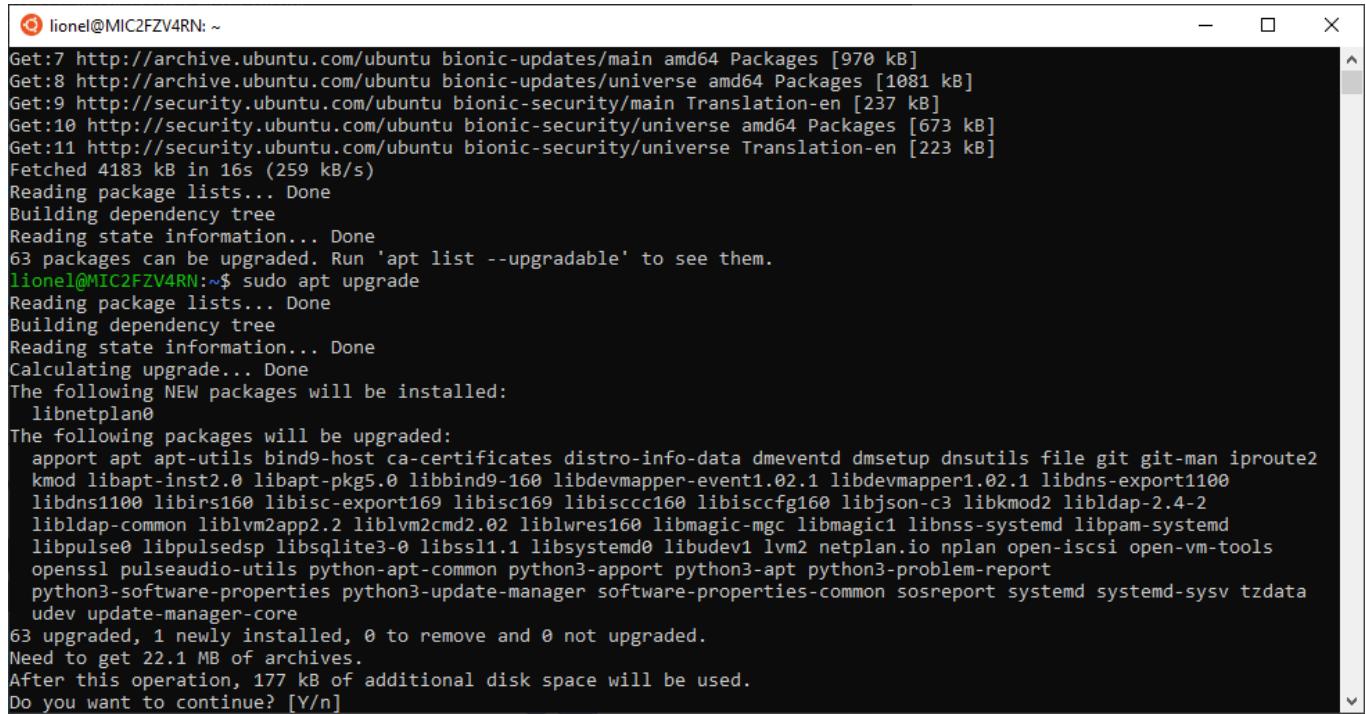


```
lionel@MIC2FZV4RN:~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: lionel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

lionel@MIC2FZV4RN:~$
```

- Notice that Python is already installed (`python3 --version`) and as it is always a good practice to update a Linux environment, enter:

```
sudo apt update  
sudo apt upgrade
```



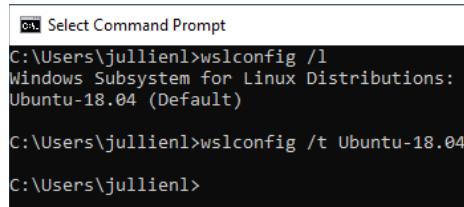
```
lione1@MIC2FZV4RN: ~  
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [970 kB]  
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1081 kB]  
Get:9 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [237 kB]  
Get:10 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [673 kB]  
Get:11 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [223 kB]  
Fetched 4183 kB in 16s (259 kB/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
63 packages can be upgraded. Run 'apt list --upgradable' to see them.  
lione1@MIC2FZV4RN:~$ sudo apt upgrade  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Calculating upgrade... Done  
The following NEW packages will be installed:  
 libnetplan0  
The following packages will be upgraded:  
 apport apt-utils bind9-host ca-certificates distro-info-data dmeventd dmsetup dnsutils file git git-man iproute2  
 kmod libapt-inst2.0 libapt-pkg5.0 libbind9-160 libdevmapper-event1.02.1 libdevmapper1.02.1 libdns-export1100  
 libdns1100 libirs160 libisc-export169 libisc169 libisccc160 libisccfg160 libjson-c3 libkmod2 libldap-2.4-2  
 libldap-common liblvm2app2.2 liblvm2cmd2.02 liblwres160 libmagic-mgc libmagic1 libnss-systemd libpam-systemd  
 libpulse0 libpulsedsp libsqlite3-0 libssl1.1 libsystemd0 libudev1 lvm2 netplan.io nplan open-iscsi open-vm-tools  
 openssl pulseaudio-utils python-apt-common python3-apport python3-apt python3-problem-report  
 python3-software-properties python3-update-manager software-properties-common sosreport systemd systemd-sysv tzdata  
 udev update-manager-core  
63 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 22.1 MB of archives.  
After this operation, 177 kB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

- When prompted, type **Yes** to install all the new packages

**Note:** Closing the WSL window does not stop Ubuntu. If you need to restart WSL Ubuntu, you need to type in a Windows command prompt:

```
wslconfig /l  
wslconfig /t Ubuntu-xx.xx
```

This command terminates the Ubuntu distribution. Once it is closed, you would need to restart Ubuntu WSL.



```
C:\Users\jullienl>wslconfig /l  
Windows Subsystem for Linux Distributions:  
Ubuntu-18.04 (Default)  
  
C:\Users\jullienl>wslconfig /t Ubuntu-18.04  
C:\Users\jullienl>
```

## Installing Ansible on Ubuntu WSL

Next, we want to install in this Linux distribution all the requirements found at <https://github.com/HewlettPackard/oneview-ansible> to run the Ansible modules for HPE OneView:

README.md

build passing coverage 100%

# Ansible Modules for HPE OneView

Modules to manage HPE OneView using Ansible playbooks.

## Requirements

- Ansible >= 2.1
- Python >= 2.7.9
- HPE OneView Python SDK

- To install Ansible, run first the following command to include the official project's PPA (personal package archive) in your system's list of sources:

```
sudo apt-add-repository ppa:ansible/ansible
```

```
lionel@MIC2FZV4RN:~$ sudo apt-add-repository ppa:ansible/ansible
Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy. Avoid writing scripts or custom code to deploy and update your applications—automate in a language that approaches plain English, using SSH, with no agents to install on remote systems.

http://ansible.com/
More info: https://launchpad.net/~ansible/+archive/ubuntu/ansible
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Ign:1 http://ppa.launchpad.net/ansible/ansible-1.9/ubuntu focal InRelease
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:5 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
```

- Press **Enter** when prompted.

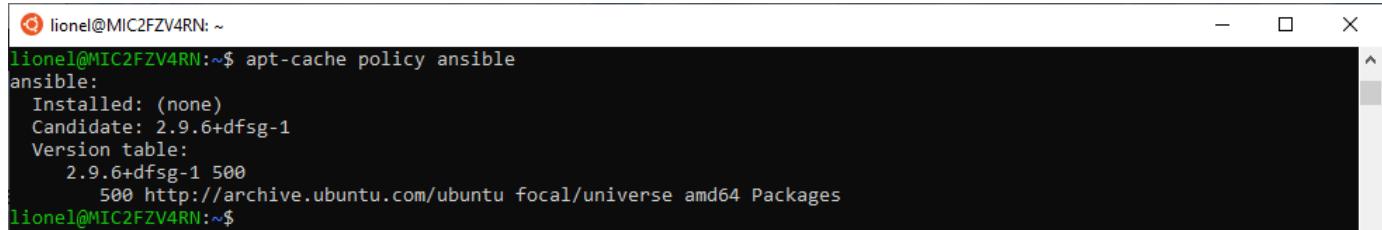
**Note:** if you are behind a corporate proxy, you need to set the proxy environment variables:

```
export http_proxy=http://proxy.myproxy.com:port
export https_proxy=https://proxy.myproxy.com:port
```

```
lionel@MIC2FZV4RN:~/oneview-python$ export http_proxy=http://web-proxy.corp.hpecorp.net:8088
lionel@MIC2FZV4RN:~/oneview-python$ export https_proxy=https://web-proxy.corp.hpecorp.net:8088
lionel@MIC2FZV4RN:~/oneview-python$
```

- To check all available packages in the repository, enter:

```
apt-cache policy ansible
```

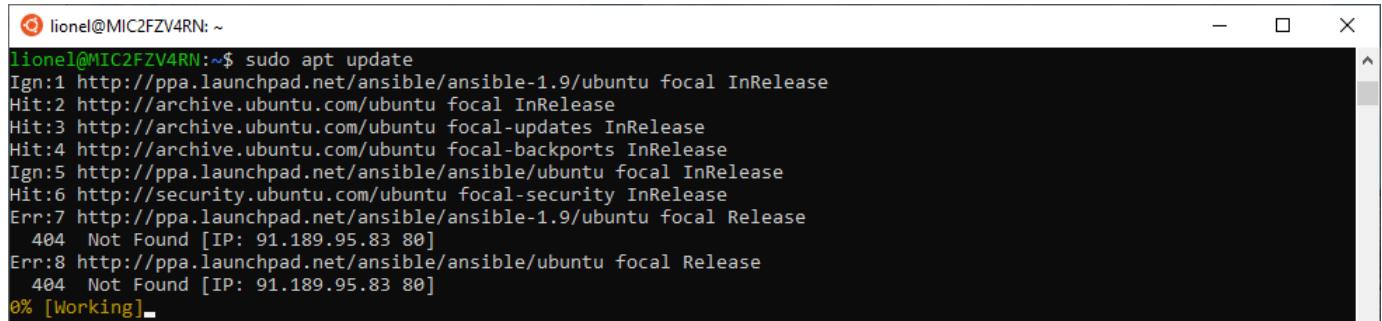


```
lionel@MIC2FZV4RN:~$ apt-cache policy ansible
ansible:
  Installed: (none)
  Candidate: 2.9.6+dfsg-1
  Version table:
    2.9.6+dfsg-1 500
      500 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages
lionel@MIC2FZV4RN:~$
```

A terminal window titled 'lionel@MIC2FZV4RN:~' showing the output of the 'apt-cache policy ansible' command. The output shows that there are no installed packages, but a candidate version 2.9.6+dfsg-1 is available from the focal/universe repository.

- Next, refresh your system's package index so that it is aware of the packages available in the newly included PPA:

```
sudo apt update
```



```
lionel@MIC2FZV4RN:~$ sudo apt update
Ign:1 http://ppa.launchpad.net/ansible/ansible-1.9/ubuntu focal InRelease
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:5 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Err:7 http://ppa.launchpad.net/ansible/ansible-1.9/ubuntu focal Release
  404  Not Found [IP: 91.189.95.83 80]
Err:8 http://ppa.launchpad.net/ansible/ansible/ubuntu focal Release
  404  Not Found [IP: 91.189.95.83 80]
0% [Working]
```

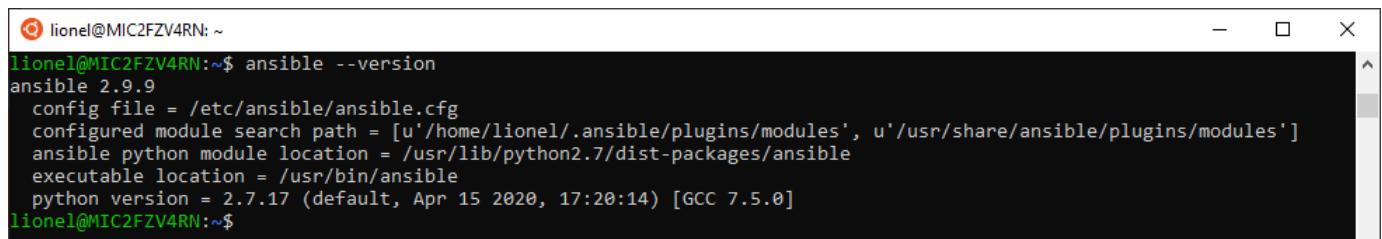
A terminal window titled 'lionel@MIC2FZV4RN:~' showing the output of the 'sudo apt update' command. It lists several package sources being checked, including the Ansible PPA and the Ubuntu focal release. There are some errors related to the Ansible PPA's Release file.

- Then we can install Ansible with:

```
sudo apt install ansible
```

- Once completed, you can check the Ansible installation by entering:

```
ansible --version
```



```
lionel@MIC2FZV4RN:~$ ansible --version
ansible 2.9.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/lionel/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Apr 15 2020, 17:20:14) [GCC 7.5.0]
lionel@MIC2FZV4RN:~$
```

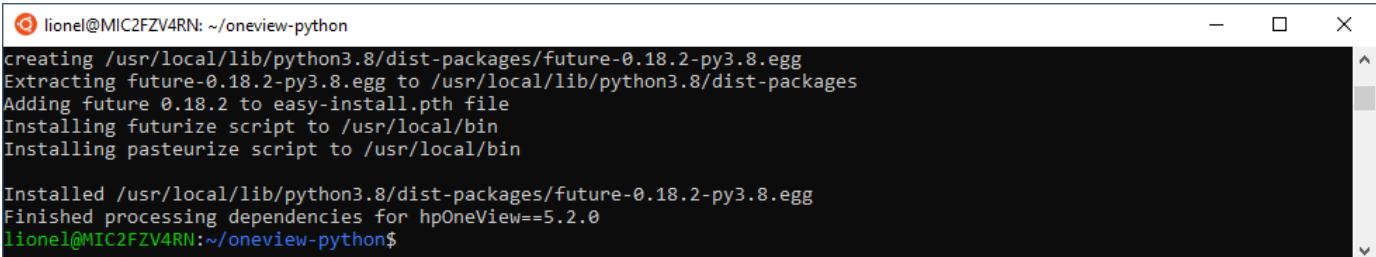
A terminal window titled 'lionel@MIC2FZV4RN:~' showing the output of the 'ansible --version' command. It displays the version of Ansible installed (2.9.9), the configuration file used, and the location of the Python module and executable.

## Installing HPE OneView Python SDK on Ubuntu WSL

To install the HPE OneView Python SDK, we are using the information from <https://github.com/HewlettPackard/oneview-python>

- Type:

```
cd ~  
git clone https://github.com/HewlettPackard/oneview-python.git  
cd oneview-python  
sudo python3 setup.py install
```



```
lionel@MIC2FZV4RN: ~/oneview-python  
creating /usr/local/lib/python3.8/dist-packages/future-0.18.2-py3.8.egg  
Extracting future-0.18.2-py3.8.egg to /usr/local/lib/python3.8/dist-packages  
Adding future 0.18.2 to easy-install.pth file  
Installing futurize script to /usr/local/bin  
Installing pasterize script to /usr/local/bin  
  
Installed /usr/local/lib/python3.8/dist-packages/future-0.18.2-py3.8.egg  
Finished processing dependencies for hpOneView==5.2.0  
lionel@MIC2FZV4RN:~/oneview-python$
```

**Note:** if you need to upgrade from a previous version of the HPE OneView Python SDK, refer to the [Upgrading your HPE Synergy demonstration environment](#) chapter.

- If you have a **ModuleNotFoundError: No module named 'setuptools'**, install the missing module using:

```
sudo apt-get install python3-setuptools
```

**Important notice:** Do not to use the *python-hpOneView* repository from <https://github.com/HewlettPackard/python-hpOneView> ! This is the repository to support legacy SDKs

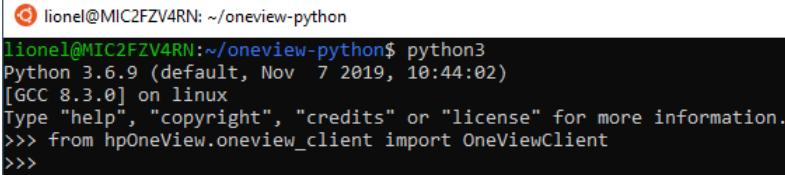
**Note:** if you are behind a corporate proxy, you need to set a Git proxy:  
`git config --global http.proxy http://proxy.myproxy.com:port/  
git config --global https.proxy http://proxy.myproxy.com:port/`

- Once the module is successfully installed, you can test the module by typing:

```
python3
```

- Then import the HPOneView module with:

```
from hpOneView.oneview_client import OneViewClient
```



```
lionel@MIC2FZV4RN: ~/oneview-python$ python3  
Python 3.6.9 (default, Nov  7 2019, 10:44:02)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from hpOneView.oneview_client import OneViewClient  
>>>
```

- Import the *pprint* function as well, we'll use it to make output more readable

```
from pprint import pprint
```

- You can examine the list of members of the *OneViewClient* class with:

```
pprint(dir(OneViewClient))
```

```
'metric_streaming',
'migratable_vc_domains',
'network_sets',
'os_deployment_plans',
'os_deployment_servers',
'power_devices',
'racks',
'restores',
'roles',
'san_managers',
'sas_interconnect_types',
'sas_interconnects',
'sas_logical_interconnect_groups',
'sas_logical_interconnects',
'sas_logical_jbod_attachments',
'sas_logical_jbods',
'scopes',
'server_hardware',
'server_hardware_types',
'server_profile_templates',
'server_profiles',
'storage_pools',
'storage_systems',
'storage_volume_attachments',
'storage_volume_templates',
'switch_types',
'switches',
'tasks',
'unmanaged_devices',
'uplink_sets',
'users',
'versions',
'veolumes']
>>>
```

If you get a response with all the list members as illustrated above, your module is successfully installed.

- To exit the Python environment, press **CTRL + z**

## Installing Ansible Modules for HPE OneView on Ubuntu WSL

Next step, we need to install the Ansible Modules for HPE OneView using the information from <https://github.com/HewlettPackard/oneview-ansible>

- First, we need to install pip. pip is the package installer for Python:

```
sudo apt-get install python3-pip
```

- Then go back to your home directory (/home/<user>):

```
cd ~
```

- Then use *git clone* to clone the Ansible Modules for HPE OneView repository:

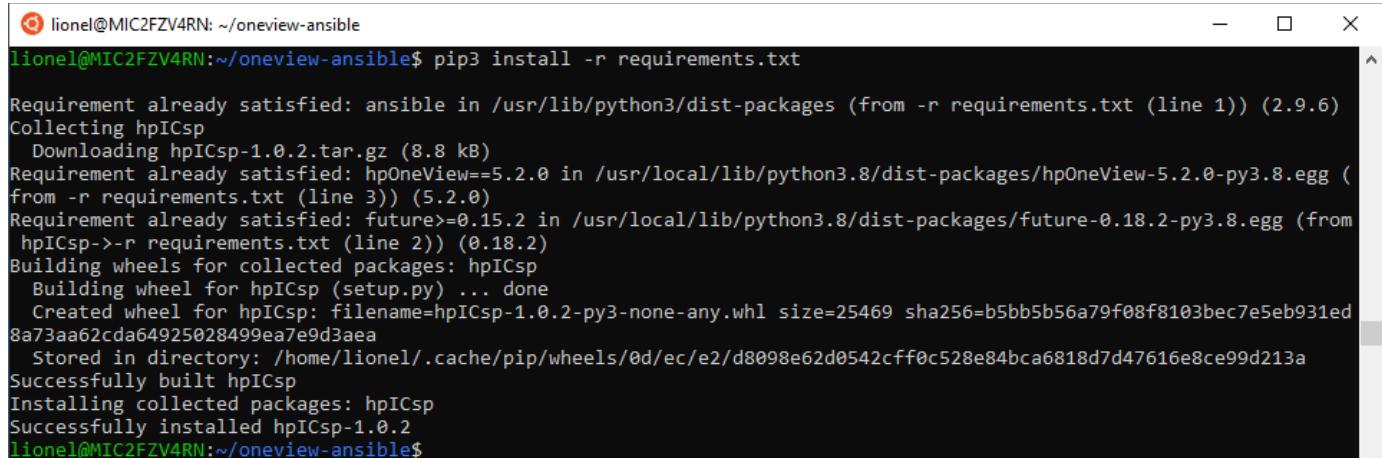
```
git clone https://github.com/HewlettPackard/oneview-ansible.git
```

**Note:** if you are behind a corporate proxy, you need to set a Git proxy:

```
git config --global http.proxy http://proxy.myproxy.com:port/
git config --global https.proxy http://proxy.myproxy.com:port/
```

- Then install the dependency packages:

```
cd oneview-ansible
pip3 install -r requirements.txt
```



```
lionel@MIC2FZV4RN: ~/oneview-ansible
lionel@MIC2FZV4RN:~/oneview-ansible$ pip3 install -r requirements.txt

Requirement already satisfied: ansible in /usr/lib/python3/dist-packages (from -r requirements.txt (line 1)) (2.9.6)
Collecting hpICsp
  Downloading hpICsp-1.0.2.tar.gz (8.8 kB)
Requirement already satisfied: hpOneView==5.2.0 in /usr/local/lib/python3.8/dist-packages/hpOneView-5.2.0-py3.8.egg (from -r requirements.txt (line 3)) (5.2.0)
Requirement already satisfied: future>=0.15.2 in /usr/local/lib/python3.8/dist-packages/future-0.18.2-py3.8.egg (from hpICsp->-r requirements.txt (line 2)) (0.18.2)
Building wheels for collected packages: hpICsp
  Building wheel for hpICsp (setup.py) ... done
  Created wheel for hpICsp: filename=hpICsp-1.0.2-py3-none-any.whl size=25469 sha256=b5bb5b56a79f08f8103bec7e5eb931ed
8a73aa62cda64925028499ea7e9d3aea
  Stored in directory: /home/lionel/.cache/pip/wheels/0d/ec/e2/d8098e62d0542cff0c528e84bca6818d7d47616e8ce99d213a
Successfully built hpICsp
Installing collected packages: hpICsp
Successfully installed hpICsp-1.0.2
lionel@MIC2FZV4RN:~/oneview-ansible$
```

**Note:** if you are behind a corporate proxy, you need to define a proxy:

```
pip3 install -r requirements.txt --proxy http://proxy.server.com:port/
```

**Note:** if you need to upgrade from a previous version of the Ansible modules for OneView, refer to the [Upgrading your HPE Synergy demonstration environment](#) chapter.

At the time of writing this document, the fix for the `oneview_server_profile` module that cannot create multiple server profiles in parallel (see <https://github.com/HewlettPackard/oneview-ansible/issues/313>) was not included in the latest release. As we need this fix for one of our demos, we must merge the branch with the fix to our cloned directory.

- We currently have only one Master branch:

```
git branch
```

```
lionel@MIC2FZV4RN:~/oneview-ansible$ git branch
* master
lionel@MIC2FZV4RN:~/oneview-ansible$
```

- To navigate to the fix branch, enter:

```
git checkout bug_fix/create_profiles_in_parallel
```

```
lionel@MIC2FZV4RN:~/oneview-ansible$ git checkout bug_fix/create_profiles_in_parallel
Branch 'bug_fix/create_profiles_in_parallel' set up to track remote branch 'bug_fix/create_profiles_in_parallel' from 'origin'.
Switched to a new branch 'bug_fix/create_profiles_in_parallel'
lionel@MIC2FZV4RN:~/oneview-ansible$
```

- This changes the active branch to the new branch. Note that the asterisk shows the currently active branch.

```
git branch
```

```
lionel@MIC2FZV4RN:~/oneview-ansible$ git branch
* bug_fix/create_profiles_in_parallel
  master
lionel@MIC2FZV4RN:~/oneview-ansible$
```

- Then to change the active branch back to `master`, run:

```
git checkout master
```

Then to merge the `bug_fix` features into the `master` branch, run:

```
git merge bug_fix/create_profiles_in_parallel
```

`git merge` merges the specified branch into the currently active branch, so we need to be on the branch that we are merging into.

**Note:** If you get an ‘empty ident name not allowed’ error when running the `git merge` command, then you need to set a git account, it could be a fake one like:

```
git config --global user.name "hpe"
git config --global user.email "hpe@synergy.lab"
```



- The following nano editor should open:

```
Merge branch 'master' into bug_fix/create_profiles_in_parallel

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text  
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^[ Go To Line M-B Redo M-6 Copy Text

- Just save the commit information by pressing **CTRL + O** then press **Enter** to save the file.
- Then press **CTRL + X** to exit the editor.

From the output, we can see that two files have been changed to implement the new fixes from the *bug\_fix* branch:

```
lionel@MIC2FZV4RN:~/oneview-ansible$ git merge bug_fix/create_profiles_in_parallel
Auto-merging library/oneview_server_profile.py
Auto-merging library/module_utils/oneview.py
Merge made by the 'recursive' strategy.
 library/module_utils/oneview.py | 1 +
 library/oneview server profile.py | 5 +++--
 2 files changed, 4 insertions(+), 2 deletions(-)
lionel@MIC2FZV4RN:~/oneview-ansible$
```

- If you type now

```
git checkout master
```

You should see that our *master* branch is now ahead of the origin branch by 2 commits:

```
lionel@MIC2FZV4RN:~/oneview-ansible$ git checkout master
Already on 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
lionel@MIC2FZV4RN:~/oneview-ansible$
```

We are good now in term of content and bug fixes, we can now continue with the Ansible module configuration.

The next step is to set persisting environment variables, let us modify the Ubuntu user profile.

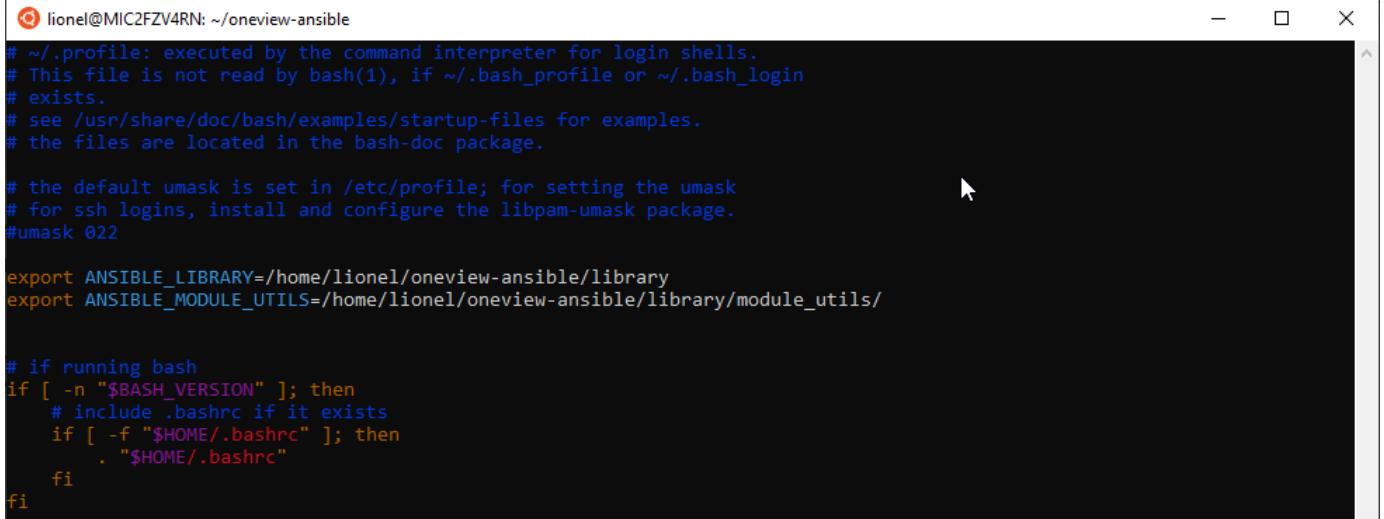
- Type:

```
vi ~/.profile
```

- Then add the following lines (press the letter **i** to put the VI editor in *Insert Mode* then copy/paste, then press **ESC** to exit *Insert Mode*, then type **:**(colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi):

```
export ANSIBLE_LIBRARY=/home/<username>/oneview-ansible/library
export ANSIBLE_MODULE_UTILS=/home/<username>/oneview-ansible/library/module_utils/
```

**Note:** Make sure you modify `<username>` with the username you have defined.



```
lionel@MIC2FZV4RN: ~/oneview-ansible
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

export ANSIBLE_LIBRARY=/home/lionel/oneview-ansible/library
export ANSIBLE_MODULE_UTILS=/home/lionel/oneview-ansible/library/module_utils/

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi
```

- To immediately apply the new environment variables, type:

```
source ~/.profile
```

**Note:** You can use **env** to check that the new environment variables have been applied

```
lionel@MIC2FZV4RN:~/oneview-ansible$ env
SHELL=/bin/bash
WSL_DISTRO_NAME=Ubuntu
ANSIBLE_LIBRARY=/home/lionel/oneview-ansible/library
NAME=MIC2FZV4RN
PWD=/home/lionel/oneview-ansible
LOGNAME=lionel
MOTD_SHOWN=update-motd
HOME=/home/lionel
LANG=C.UTF-8
LS_COLORS=r=0:di=0;34:ln=0;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd=40;33
30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31
31:*.lza=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=01;31
*:gz=01;31:*.lrz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.tbz=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sa
ace=01;31:*.zoo=01;31:*.cpio=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swr
pg=01;35:*.jpeg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*
*:xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;
01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;
1;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35
36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36
36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
LESSCLOSE=/usr/bin/lesspipe %s %
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %
USER=lionel
ANSIBLE_MODULE_UTILS=/home/lionel/oneview-ansible/library/module_utils/
SHLVL=1
WSLENV=
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
```

- Then finally configure the OneView appliance connection settings that will be used by the Ansible Modules for HPE OneView. Go back to your user home directory:

```
cd ~
```

- Then create a **oneview\_config.json** file with our appliance information:

```
cat > oneview_config.json << "EOF"
{
    "ip": "192.168.56.101",
    "api_version": 1800,
    "credentials": {
        "userName": "administrator",
        "authLoginDomain": "",
        "password": "password"
    }
}
EOF
```

**Note:** 192.168.56.101 is the IP address we will configure later for the DCS appliance taken from the “Host-only” VirtualBox subnet (192.168.56.0/24). **1800** is the API version of OneView 5.30.

## Installing Terraform on Ubuntu WSL

Next, we want to install all the requirements found at <https://github.com/HewlettPackard/terraform-provider-oneview> to run the Terraform provider for HPE OneView:

A screenshot of a GitHub repository page for `terraform-provider-oneview`. The page includes a `README.md` file, a green `build passing` badge, and a brief description: "A Terraform provider for oneview".

### Installing `terraform-provider-oneview` with Go

- Install Go 1.11. For previous versions, you may have to set your `$GOPATH` manually, if you haven't done it yet.
- Install Terraform 0.11.x [from here](#) and save it into `/usr/local/bin/terraform` folder (create it if it doesn't exists). This provider DOES NOT SUPPORT Terraform 0.12 or above.
- Download the code by issuing a `go get` command.

```
# Download the source code for terraform-provider-oneview
# and build the needed binary, by saving it inside $GOPATH/bin
$ go get -u github.com/HewlettPackard/terraform-provider-oneview

# Copy the binary to have it along the terraform binary
$ mv $GOPATH/bin/terraform-provider-oneview /usr/local/bin/terraform
```

We cannot install the latest version of Terraform as `terraform-provider-oneview` does not support Terraform 0.12 and above. Therefore, we will install Terraform 0.11.14.

- To download Terraform 0.11.14. enter:

```
cd ~
wget https://releases.hashicorp.com/terraform/0.11.14/terraform_0.11.14_linux_amd64.zip
```

- To unzip the downloaded file, let us install unzip:

```
sudo apt-get install unzip
```

- Unzip then the archive in `/usr/local/bin`:

```
sudo unzip -d /usr/local/bin terraform_0.11.14_linux_amd64.zip
rm terraform_0.11.14_linux_amd64.zip
```

- To verify Terraform is installed, run:

```
terraform -v
```

```
lionel@MIC2FZV4RN: ~
lionel@MIC2FZV4RN:~$ terraform -v
Terraform v0.11.14

Your version of Terraform is out of date! The latest version
is 0.12.26. You can update by downloading from www.terraform.io/downloads.html
lionel@MIC2FZV4RN:~$
```

## Installing Go on Ubuntu WSL

Next, we need to install Go on our Ubuntu WSL. Go is a modern open-source programming language created by Google that our Terraform provider for HPE OneView requires.

- From your home folder, enter:

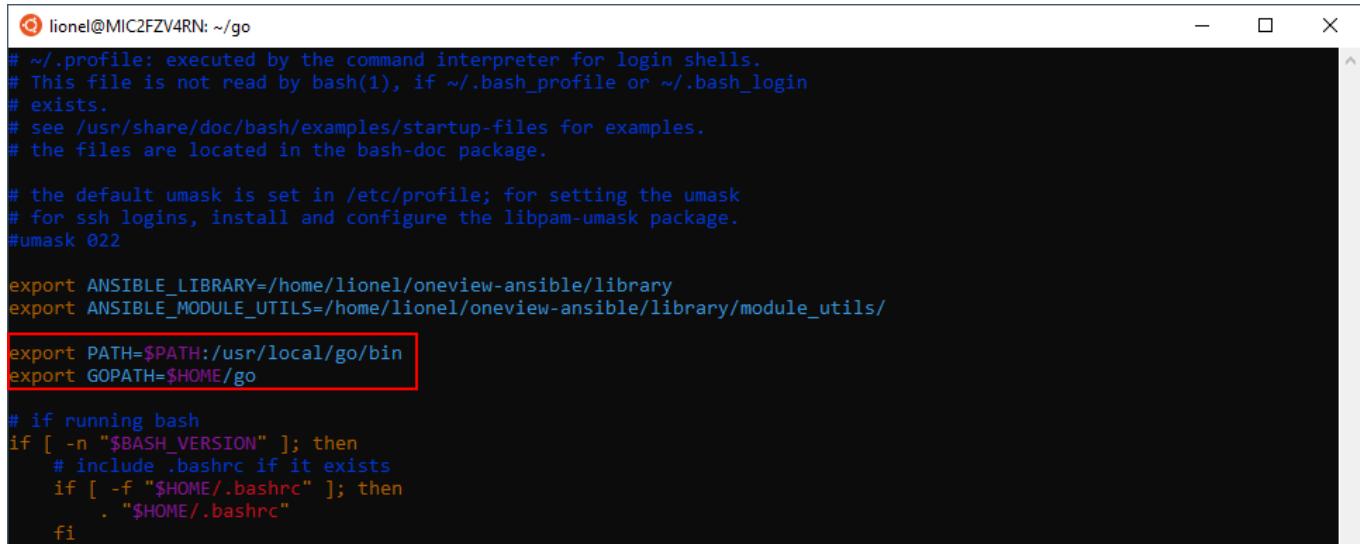
```
wget https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz  
sudo tar -C /usr/local -xzf go1.14.2.linux-amd64.tar.gz
```

- The next step is to set the \$GOPATH environment variable and add the go installation folder to \$PATH in the Ubuntu user profile. Type:

```
vi ~/.profile
```

- Then add the following line (press the letter **i** to put the VI editor in *Insert* Mode then copy/paste, then press **ESC** to exit *Insert* Mode, then type **:**(colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi):

```
export PATH=$PATH:/usr/local/go/bin  
export GOPATH=$HOME/go
```



```
lionel@MIC2FZV4RN: ~/go  
# ~/.profile: executed by the command interpreter for login shells.  
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login  
# exists.  
# see /usr/share/doc/bash/examples/startup-files for examples.  
# the files are located in the bash-doc package.  
  
# the default umask is set in /etc/profile; for setting the umask  
# for ssh logins, install and configure the libpam-umask package.  
#umask 022  
  
export ANSIBLE_LIBRARY=/home/lionel/oneview-ansible/library  
export ANSIBLE_MODULE_UTILS=/home/lionel/oneview-ansible/library/module_utils/  
  
export PATH=$PATH:/usr/local/go/bin  
export GOPATH=$HOME/go  
  
# if running bash  
if [ -n "$BASH_VERSION" ]; then  
    # include .bashrc if it exists  
    if [ -f "$HOME/.bashrc" ]; then  
        . "$HOME/.bashrc"  
    fi
```

- To immediately apply the new environment variables, type:

```
source ~/.profile
```

**Note:** Now you can type **go** to check that the command is found



## Technical white paper

```
lionel@MIC2FZV4RN: ~/terraform/hello
The commands are:

bug      start a bug report
build    compile packages and dependencies
clean    remove object files and cached files
doc      show documentation for package or symbol
env      print Go environment information
fix      update packages to use new APIs
fmt      gofmt (reformat) package sources
generate generate Go files by processing source
get      add dependencies to current module and install them
install   compile and install packages and dependencies
list     list packages or modules
mod      module maintenance
run      compile and run Go program
test     test packages
tool     run specified go tool
version  print Go version
vet      report likely mistakes in packages

Use "go help <command>" for more information about a command.

Additional help topics:

buildmode build modes
c         calling between Go and C
cache    build and test caching
environment environment variables
filetype  file types
go.mod    the go.mod file
gopath   GOPATH environment variable
gopath-get legacy GOPATH go get
goproxy   module proxy protocol
importpath import path syntax
modules   modules, module versions, and more
module-get module-aware go get
module-auth module authentication using go.sum
module-private module configuration for non-public modules
packages  package lists and patterns
testflag  testing flags
testfunc  testing functions

Use "go help <topic>" for more information about that topic.
```

- You can quickly test Go by creating a quick Go script:

```
cd ~
mkdir hello
cd hello
cat > hello.go << EOF

package main

import "fmt"

func main() {
    fmt.Printf("Hello, World\n")
}
EOF
```

- Then to translate the source code into a binary executable, enter:

```
go run hello.go
```

You should see the console displaying *Hello, World*

## Installing the Terraform provider for HPE OneView

- To install the Terraform provider for HPE OneView, enter from any folder:

```
go get -u github.com/HewlettPackard/terraform-provider-oneview
```

**Note:** if you are behind a corporate proxy, you need to set a Git proxy:

```
git config --global http.proxy http://proxy.myproxy.com:port/
```

```
git config --global https.proxy http://proxy.myproxy.com:port/
```

This command performs the following actions:

1. Builds the needed binary and save it in `/home/<username>/go/bin`  
`(= $GOPATH/bin)`
  2. Copies all source files in `home/<username>/go/src` which ends up being:  
`/home/<username>/go/src/github.com/HewlettPackard/terraform-provider-oneview/`
- The next step is to copy the generated binary into `/usr/local/bin` to have it along with the terraform binary

```
sudo mv $GOPATH/bin/terraform-provider-oneview /usr/local/bin
```

## Installing Git for VS Code source control

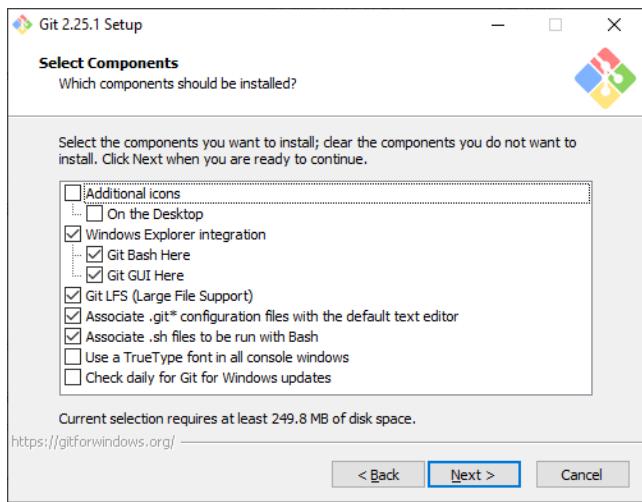
Visual Studio Code does come with an integrated Git source control provider. However, for that to work, Git itself needs to be installed on your Windows system as well.

We are going to use Git in the PowerShell project to clone GitHub repositories in our VS Code workspace but also to make sure we always have the latest content that is regularly published by the developers. For the Linux/Ansible project, we will use the Git in Ubuntu WSL.

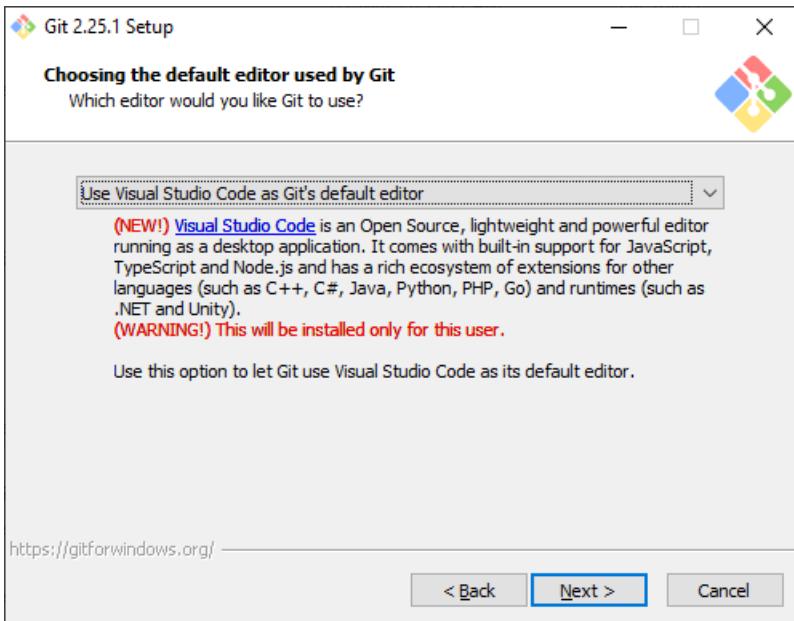
- Go to <https://git-scm.com/> and download and install Git on your machine.
- Select **Windows** and launch the installation:



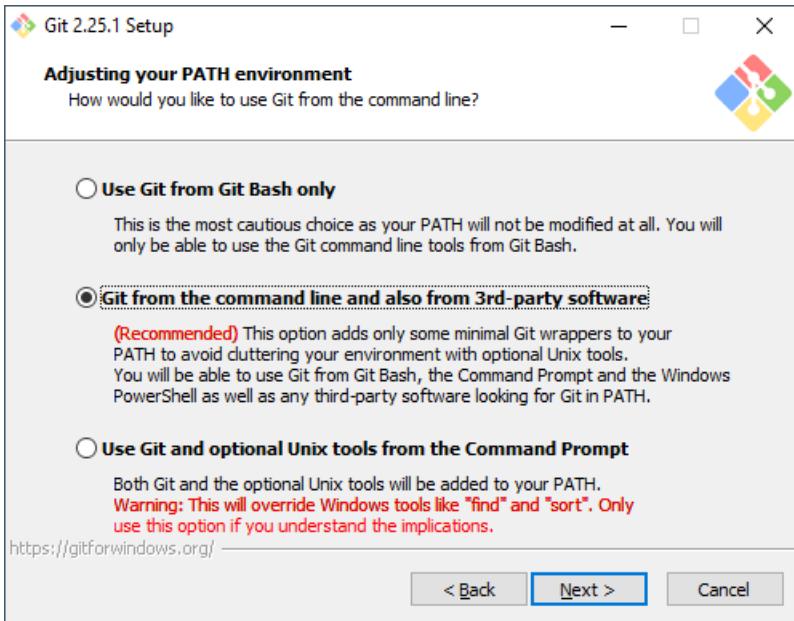
- Keep the proposed default components selected



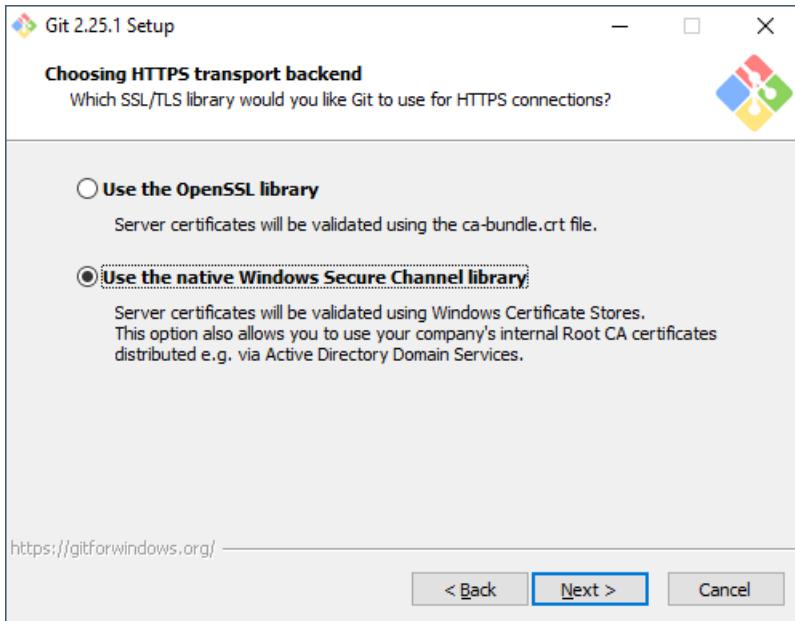
- Choose Visual Studio Code as Git's default editor option



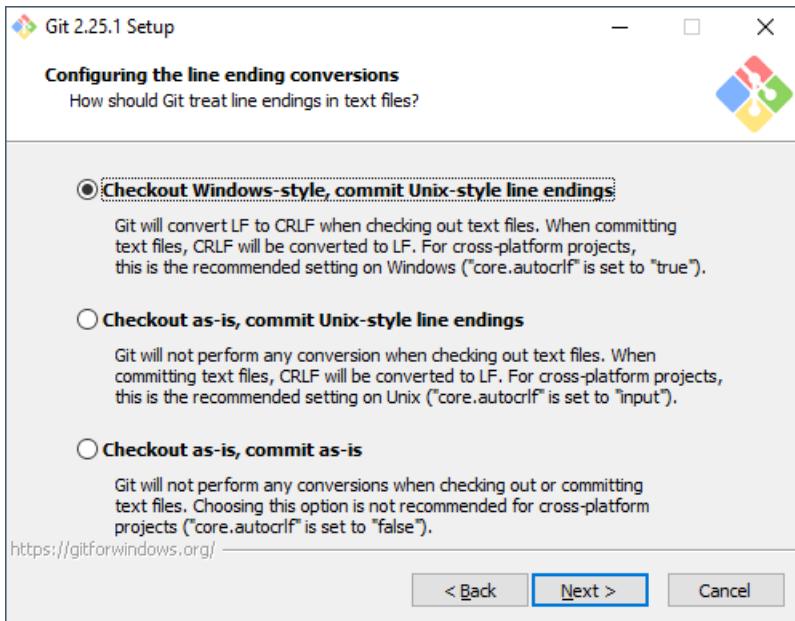
- Keep the recommended PATH environment option selected



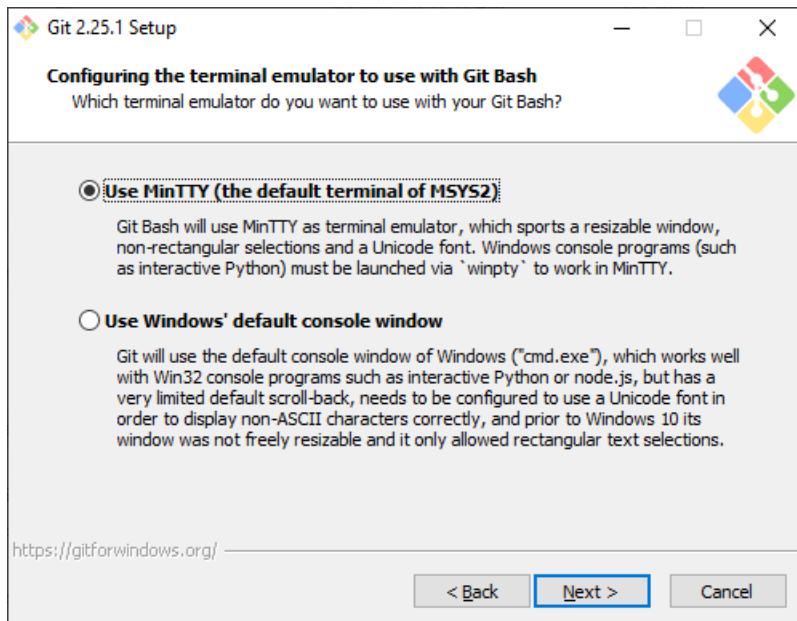
- Use the proper HTTPS transport backend option according to your environment. For HPE employees, I would recommend using the native **Windows Secure Channel library**:



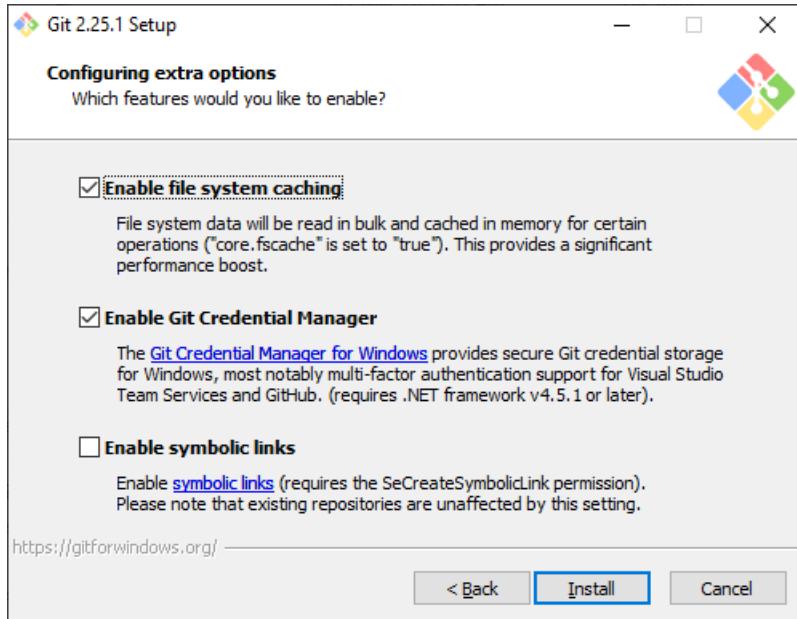
- Use the **Checkout Windows-style** as it is the recommended setting on Windows:



- Keep the default **MinTTY** option for the terminal emulator dialog



- Keep the default extra options



Once the installation is complete, restart Visual Studio Code to activate Git.

## Where do my files live in my Ubuntu WSL?

An Ubuntu WSL and a normal Linux distribution are a little different.

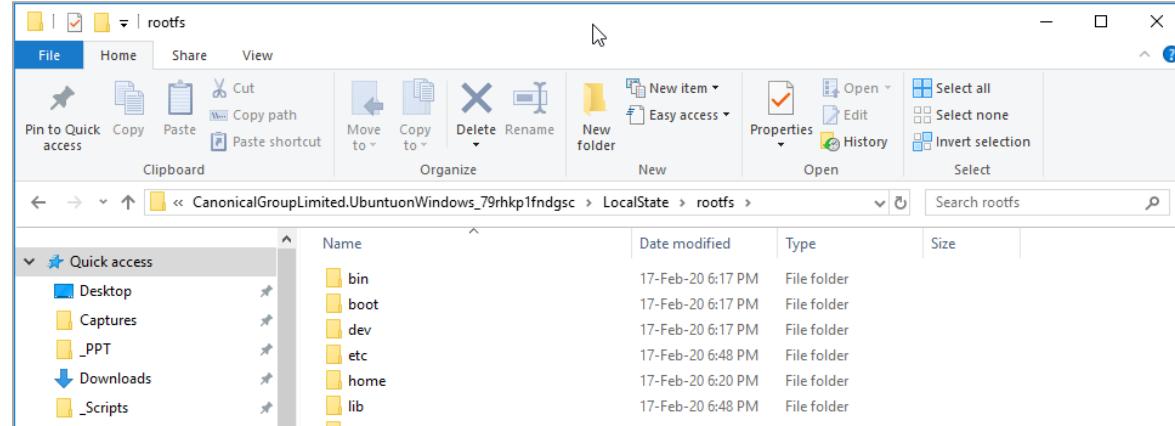
/mnt/c/ shortcut maps to c:\ Windows drive

```
lione1@MIC2FZV4RN:~/oneview-ansible$ cd /mnt/c/
$GetCurrent/          PerLogs/
$Recycle.Bin/          Program Files/
Config.Msi/            Program Files (x86)/
FFOutput/              ProgramData/
GAC_MSIL/              Quarantine/
HP/                   Recovery/
HPSDM/                SCS S-1/
HP_Color_LaserJet_Pro_MFP_M477/ SWSetup/
Intel/                SY_dcs-master/
OneDriveTemp/          SoftPaqDownloadDirectory/
                                         System Volume Information/
                                         Temp/
                                         Users/
                                         Windows/
                                         Windows.old/
                                         Windows10Upgrade/
                                         inetpub/
                                         system.sav/
```

If you want to save your work in your Windows User Home directory, you can simply save your file to the  
/mnt/c/Users/<windowsusername>/Documents

**Note:** Ubuntu root directory is located on your Windows file system in a hidden folder inside your User AppData directory:

%USERPROFILE%\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows\_xxx\LocalState\rootfs\



**Warning!** Do not open/edit any files here from Windows. Just ignore the files here. Editing any files from Windows can damage your Linux distribution.

**Note:** If you need to copy a file from your Windows file system to WSL, it is recommended to use  
cp /mnt/c/<path> /home/<username> from WSL to avoid messing up with Linux read/write permission access

This concludes Chapter-2, our Ubuntu configuration is complete and ready to be used. In the next chapter, we will install and configure VS Code.

## Chapter-3 – Preparing Microsoft Visual Studio Code

### Installing Visual Studio Code

Visual Studio Code (also known as VS Code) is one of the most popular software for source-code editing developed by Microsoft. It includes many good features for debugging, it supports embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

If you want to edit and run scripts to impress your customer, Visual Studio Code is one of the main tools you need to have in your demo toolbox.

- If not already, you can install VS Code from <https://code.visualstudio.com/Download>
- You can either select *User* or *System installer*. *System installer* does just install VS Code for all users in your system.

### Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

The screenshot shows the official download page for Visual Studio Code. At the top, there's a large "Download Visual Studio Code" button. Below it, a sub-header says "Free and built on open source. Integrated Git, debugging and extensions." There are three main download sections: one for Windows (Windows 7, 8, 10), one for Linux (.deb for Debian, Ubuntu; .rpm for Red Hat, Fedora, SUSE), and one for Mac (macOS 10.10+). Each section has a large platform-specific icon (Windows logo, Tux, Apple logo). Below each section, there are two rows of download links: "User Installer" (Windows: 64 bit, 32 bit; Linux: .deb, .rpm, .tar.gz; Mac: 64 bit) and "System Installer" (Windows: 64 bit, 32 bit; Linux: .zip; Mac: 64 bit, 32 bit). A "Snap Store" link is also present.

Platform	User Installer	System Installer
Windows	64 bit 32 bit	64 bit 32 bit
Linux	.deb .rpm .tar.gz	64 bit 64 bit 64 bit
Mac	macOS 10.10+	64 bit 32 bit

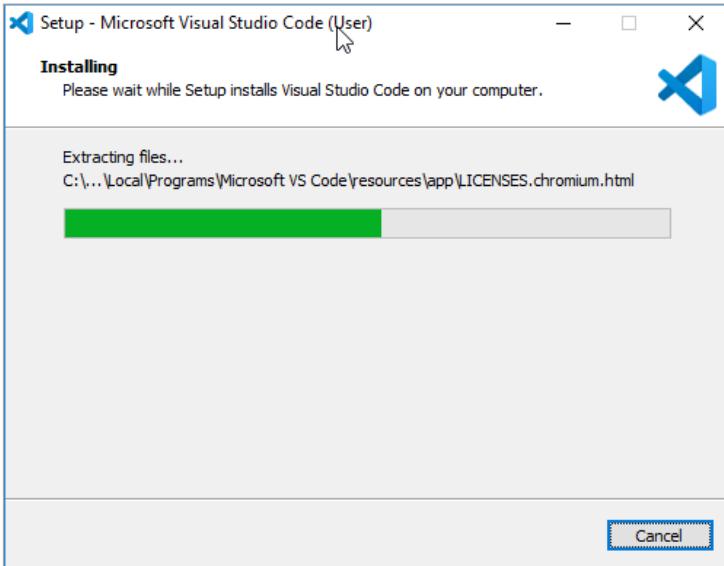
[User Installer](#)   [.deb](#)   [.rpm](#)   [.tar.gz](#)   [macOS 10.10+](#)

[System Installer](#)   [64 bit](#)   [32 bit](#)   [.zip](#)   [64 bit](#)   [32 bit](#)

[.deb](#)   [64 bit](#)  
[.rpm](#)   [64 bit](#)  
[.tar.gz](#)   [64 bit](#)

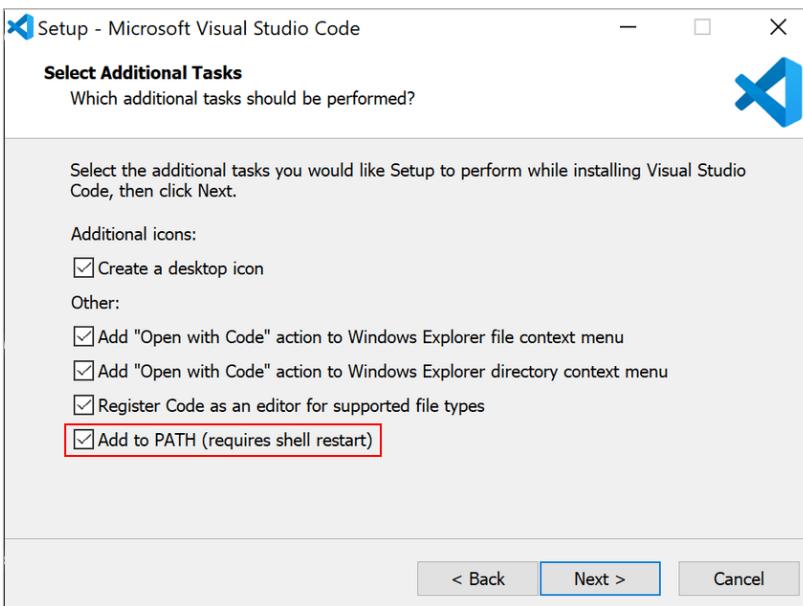
[Snap Store](#)

- You can proceed with the installation using all default parameters.



**Note:** If you need to learn more about VS Code, you can watch the introductory videos available at <https://code.visualstudio.com/docs/getstarted/introvideos> or read the Next Steps section at [https://code.visualstudio.com/docs/setup/windows#\\_next-steps](https://code.visualstudio.com/docs/setup/windows#_next-steps)

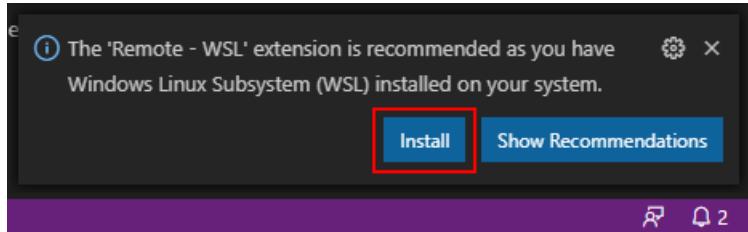
- When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily interact with WSL.



- When completed, launch **VS Code**.

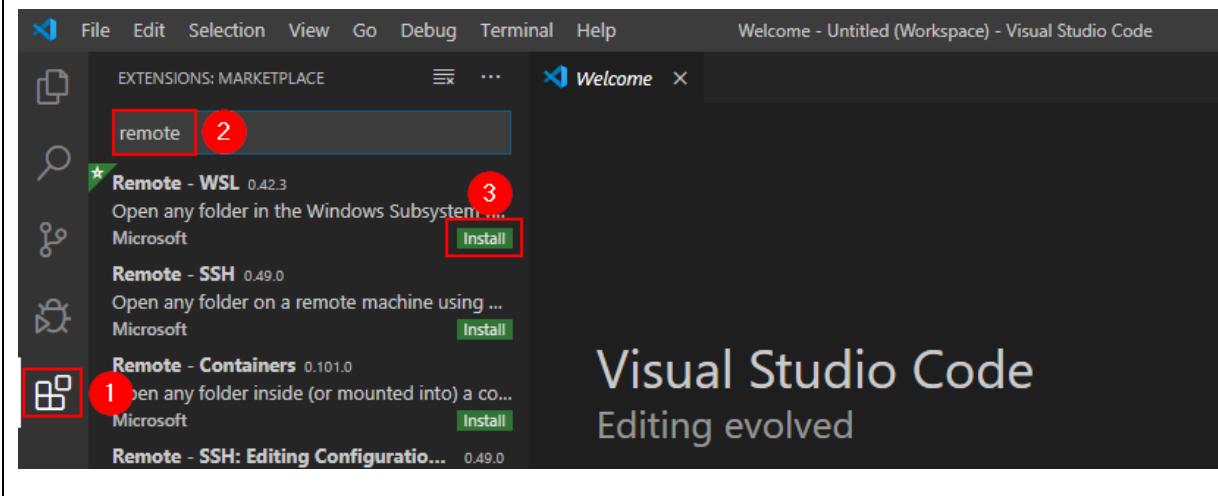
When VS Code is opening, you should notice a warning message saying "The 'Remote - WSL' extension is recommended as you have Windows Linux Subsystem (WSL) installed on your system"

- Click on **Install**



This is the extension that VS Code need to connect to our Ubuntu running in the Windows Subsystem for Linux.

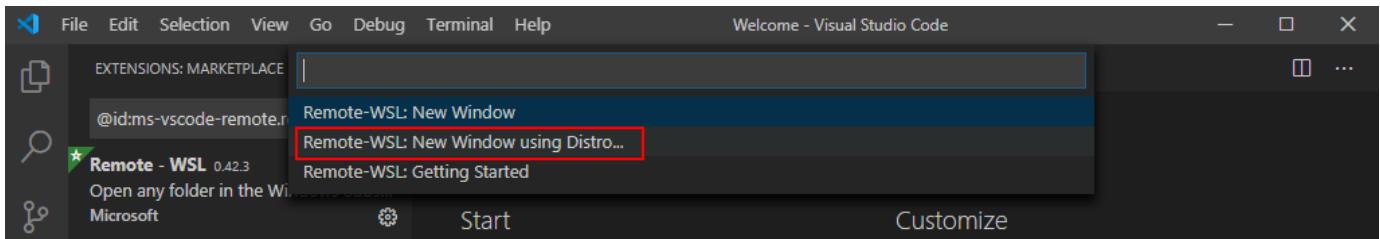
**Note:** If the message is not appearing, you can install manually this extension by clicking on the **Extensions** icon on the Activity bar and search for **Remote - WSL**



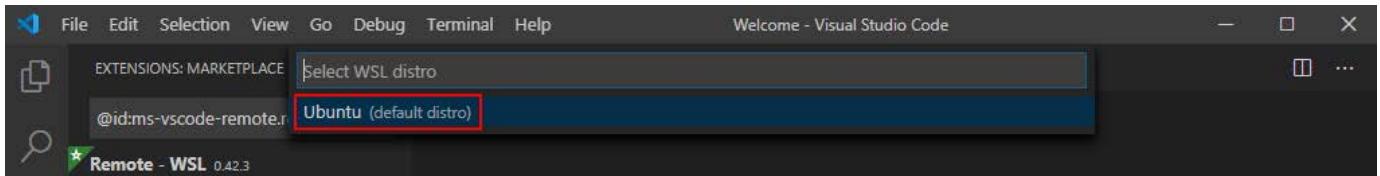
- Once installation complete, click on the Remote "Quick Access" status bar item in the lower left corner to get a list of the most common commands.



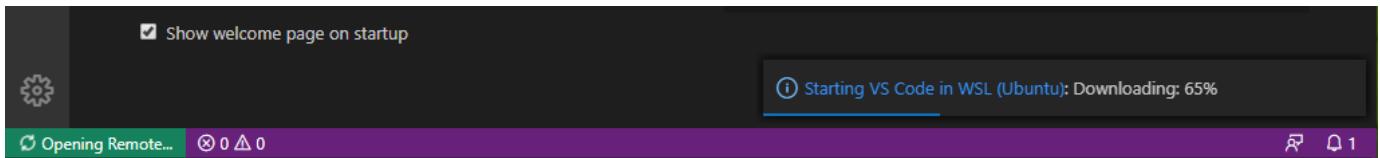
- Select **Remote-WSL: New Window using Distro...**



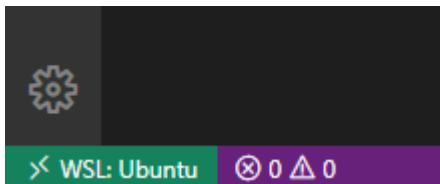
- Then select **Ubuntu**



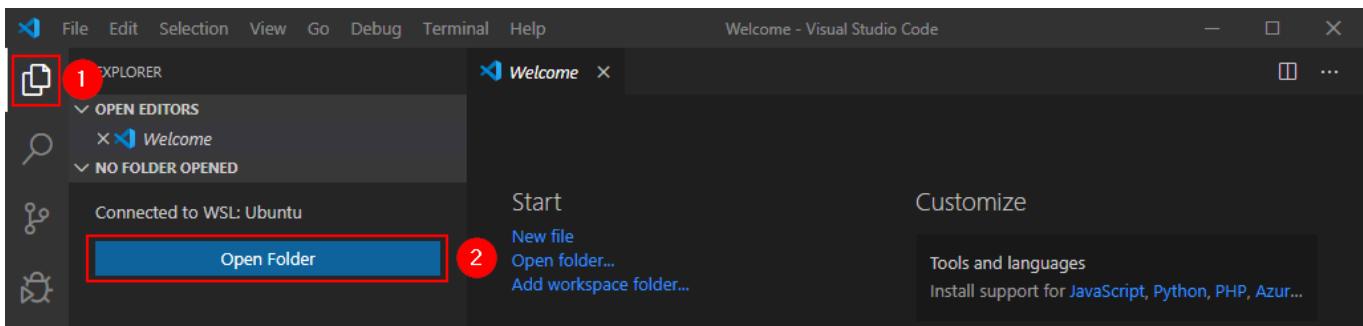
- A Remote connection to Ubuntu is taking place in a new VS Code window:



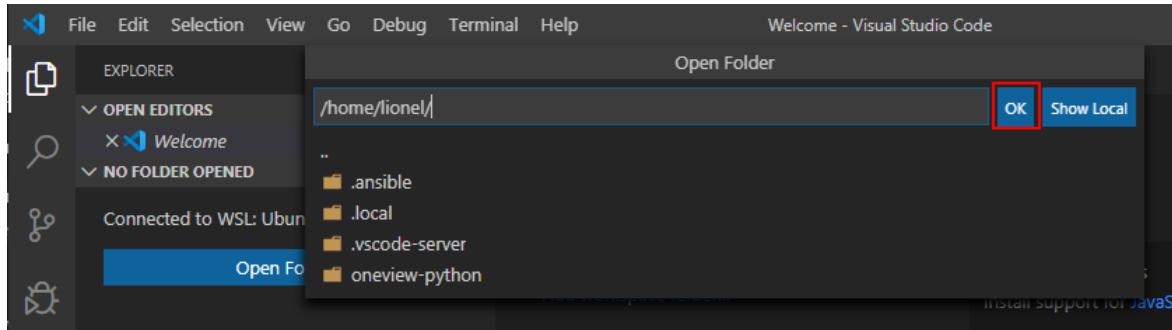
- You can close the previous VS Code window.
- Once connected, in the lower left corner of the Status Bar, you should see that you're connected to your WSL: Ubuntu instance.



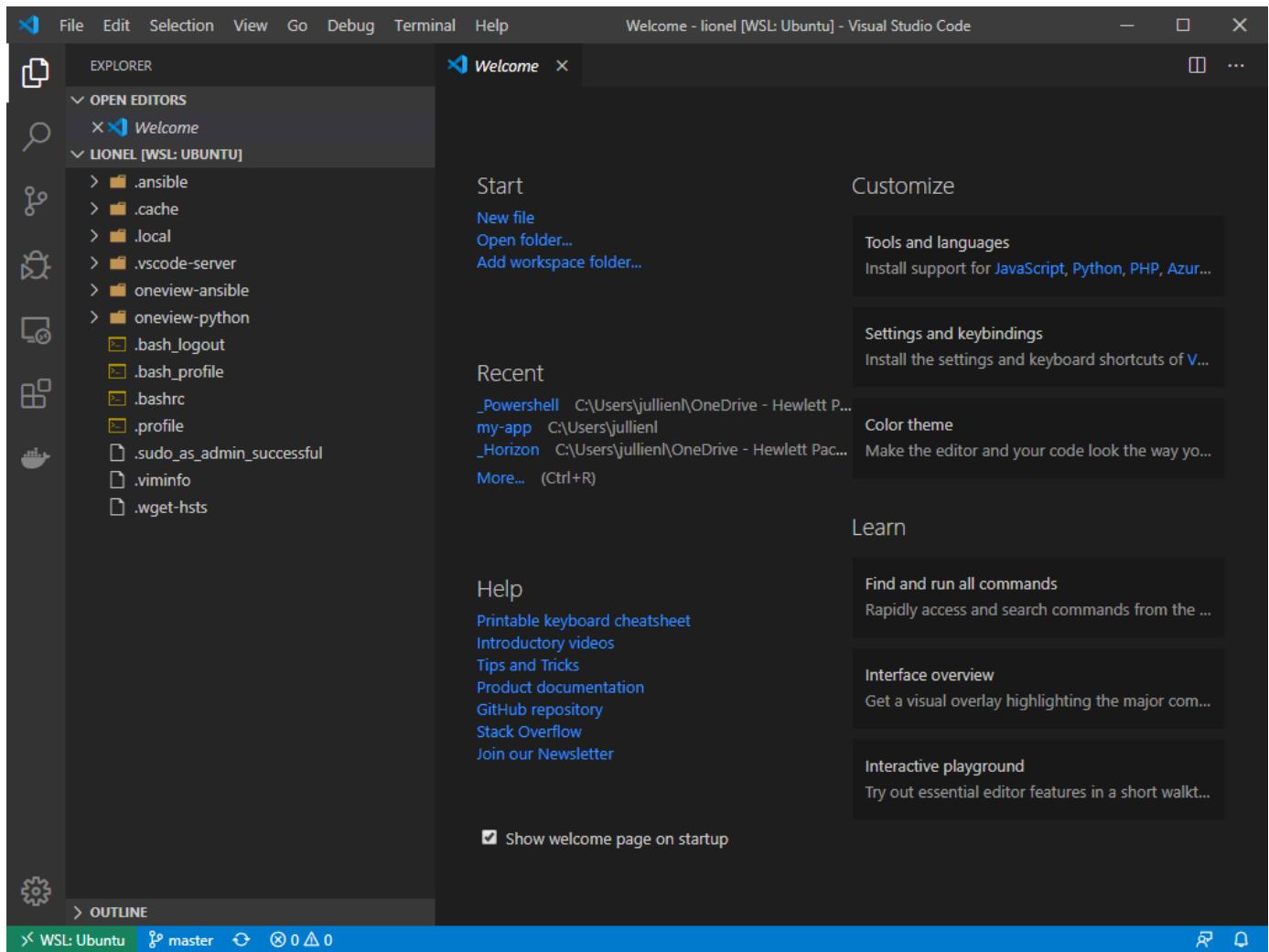
- You can then open a folder located on the WSL:Ubuntu using the **Explorer** icon on the Activity bar



- Click **OK** to the `/home/<username>` folder



- The Ubuntu user home directory content is now available in the Explorer pane:

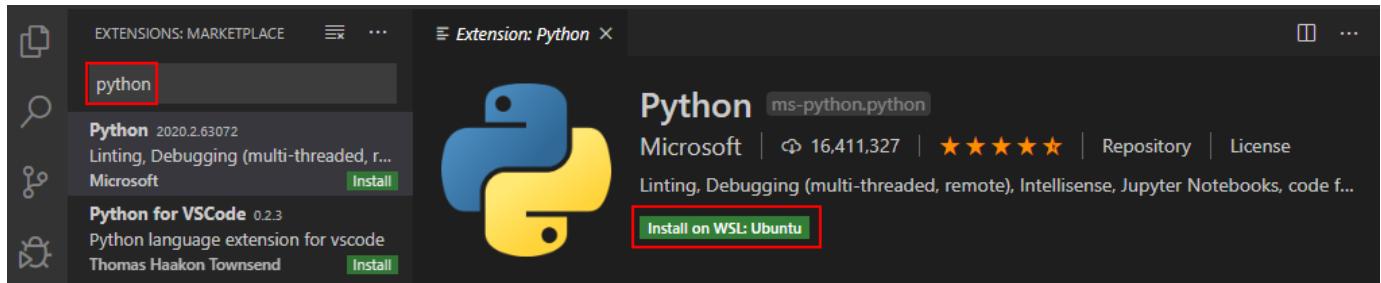


## Preparing VS Code for Python

You can extend the VS Code editor experience using tons of extensions. The VS Code community has built hundreds of useful extensions available on the VS Code Marketplace.

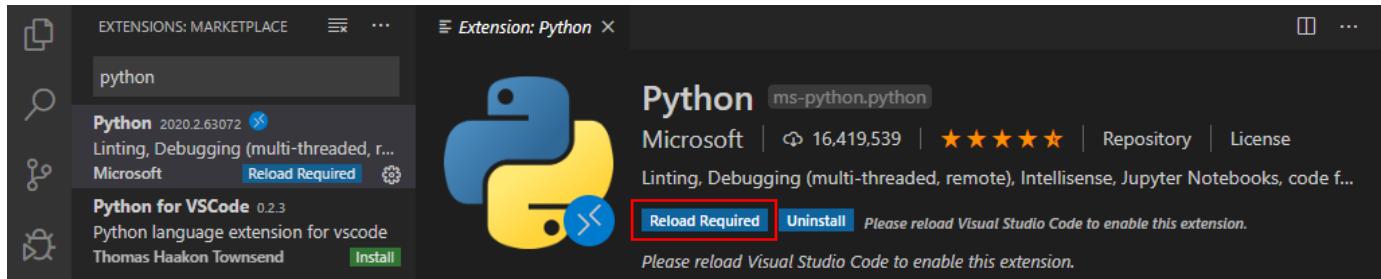
There are popular ones for PowerShell, Python, Ansible...

- Click on **Extensions** on the Activity bar, and search for **Python** and click on **Install on WSL: Ubuntu**



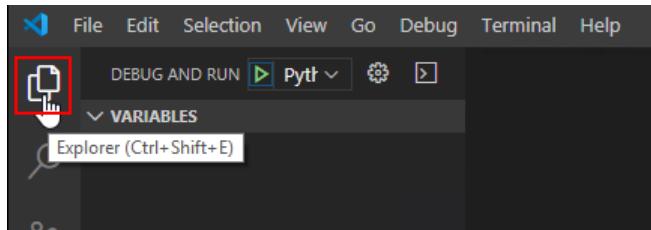
This extension provides nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to module documentation, ability to run the active script file in the VS Code terminal console, etc.

- Once the installation is complete, click on **Reload Required** to activate the extension.

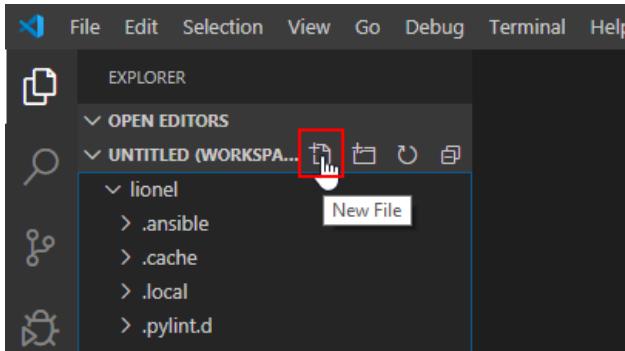


**Note:** For the Python extension, we are going to use the Python interpreter running in our Ubuntu Linux distribution hosted by WSL. Another good option if you don't want to use WSL would be to run Python directly on your Windows system, see <https://www.python.org/downloads/windows/> and <https://docs.microsoft.com/en-us/windows/python/beginners>

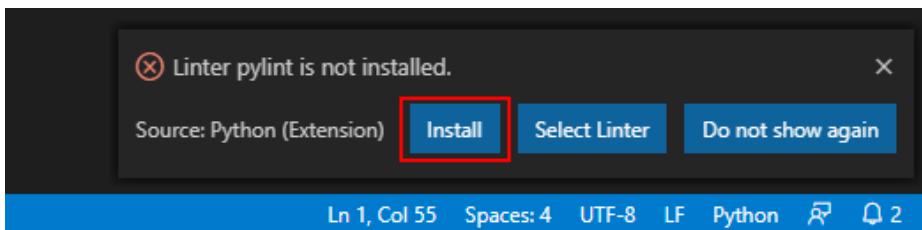
- Let us now test the Python interpreter, click now on **Explorer** on the Activity Bar



- Then click on **New File**



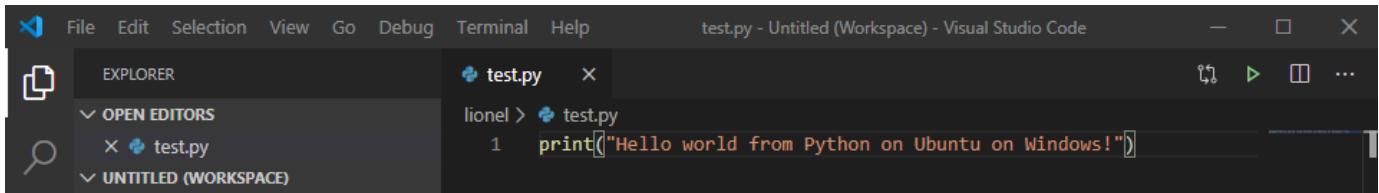
- Enter a name like **test.py** then press **ENTER**
- You can install pylint by clicking on **Install** when the window pops-up at the creation of the Python script. A code linter is a program that analyses your source code for potential errors and code style guideline violations. Pylint is one of the well-known Python linter programs similar to Pychecker and Pyflakes.



**Note:** Behind a proxy, make sure you add `--proxy` to the Pylint install command:  
`/usr/bin/python3 -m pip install -U pylint --user --proxy http://proxy.net:8088`

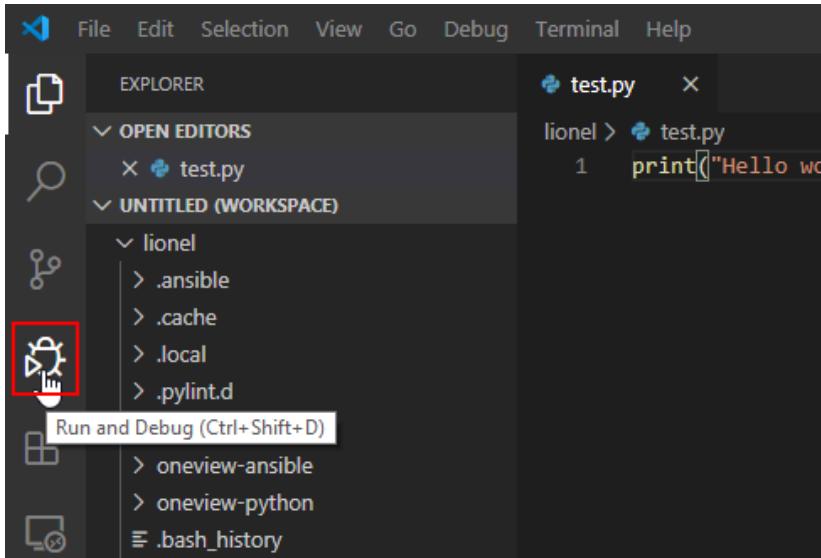
- Enter the following line to build the Python script test.

```
print("Hello world from Python on Ubuntu on Windows!")
```

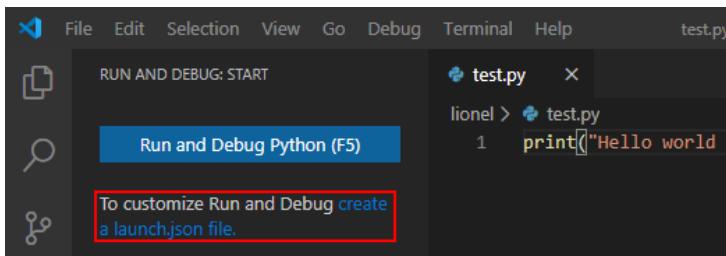


- Save the file by pressing **CTRL + S**

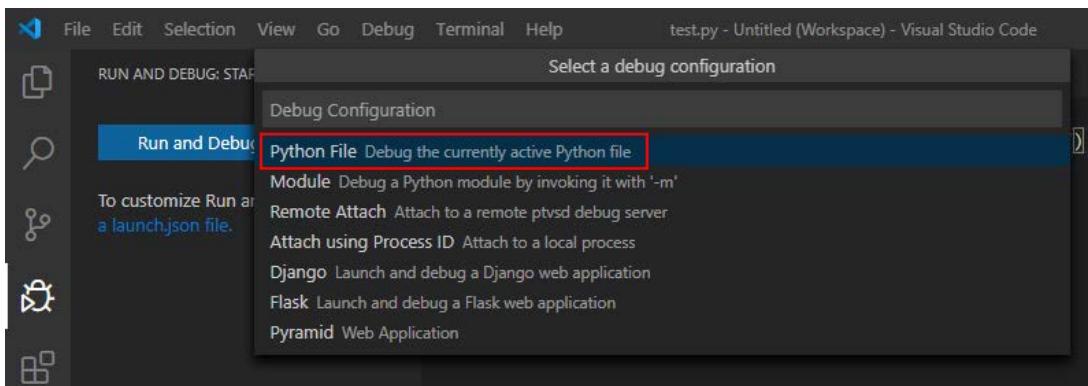
- Click on **Run and Debug**



- Select **Customize Run and Debug create a launch.json file**



- Select **Python File**

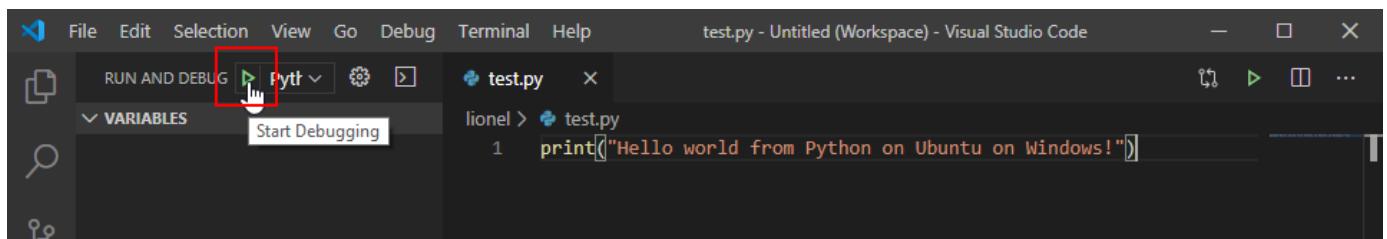


- To complete the Python debug and Run configuration, press **CTRL + S** to save the configuration file then **CTRL + F4** to close the configuration file.

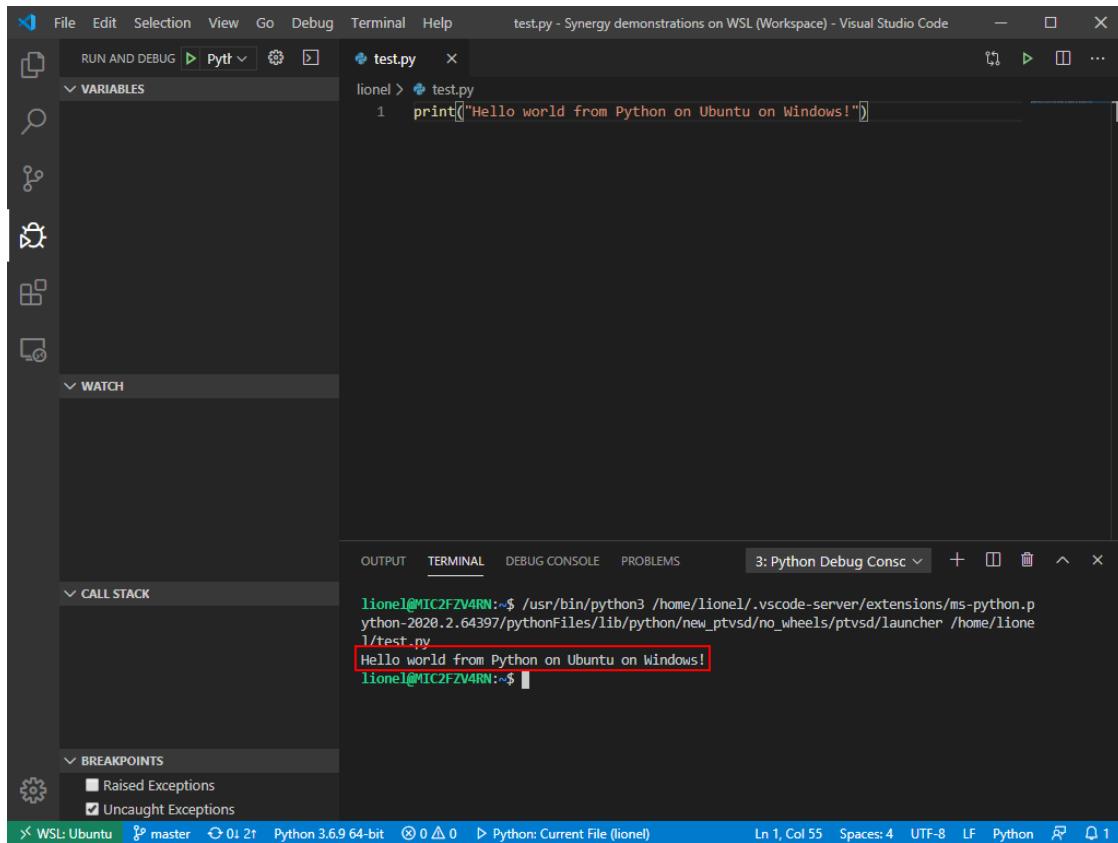
**Tip:** You can see the Python interpreter version in use in the status bar



- Now click on **Start Debugging**



- The VS Code console should display the *Hello world* message:

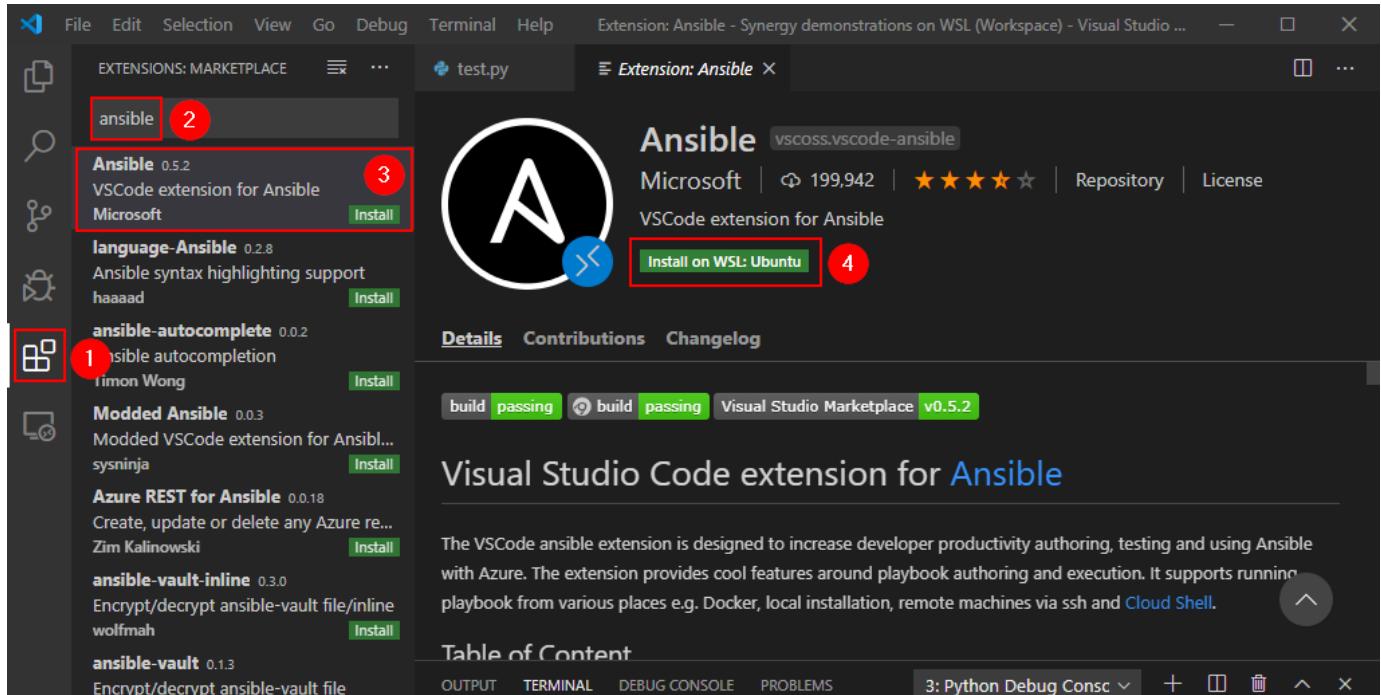


We are now all set for Python scripting and debugging.

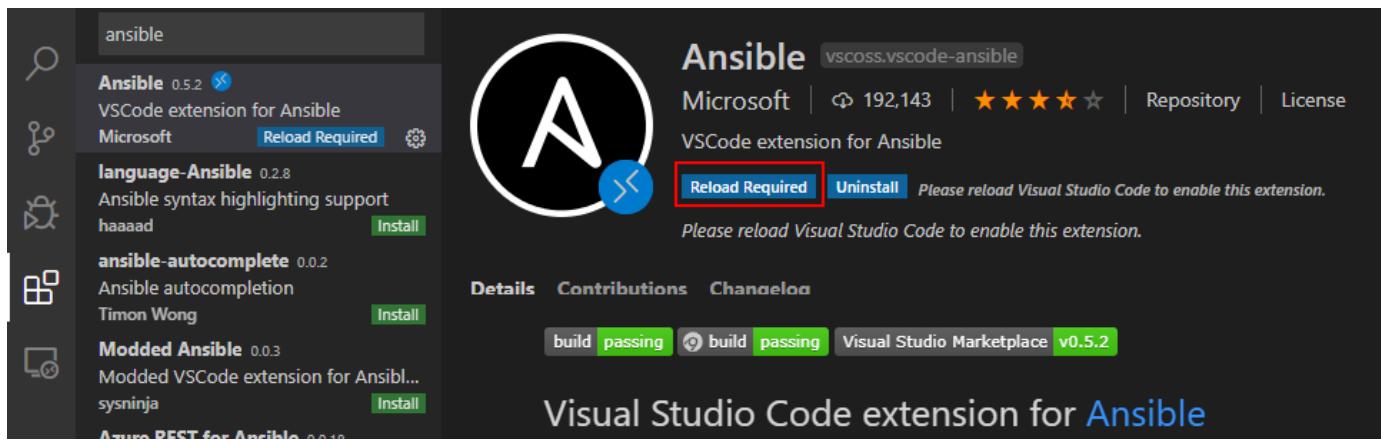
## Preparing VS Code for Ansible

To extend our VS Code editor experience with Ansible, we are going to install an Ansible extension. This extension provides cool features around playbook authoring and execution and more importantly, it supports running playbook from the WSL Ubuntu instance.

- Go to the **Extensions** and search for the **Ansible** extension and install it.



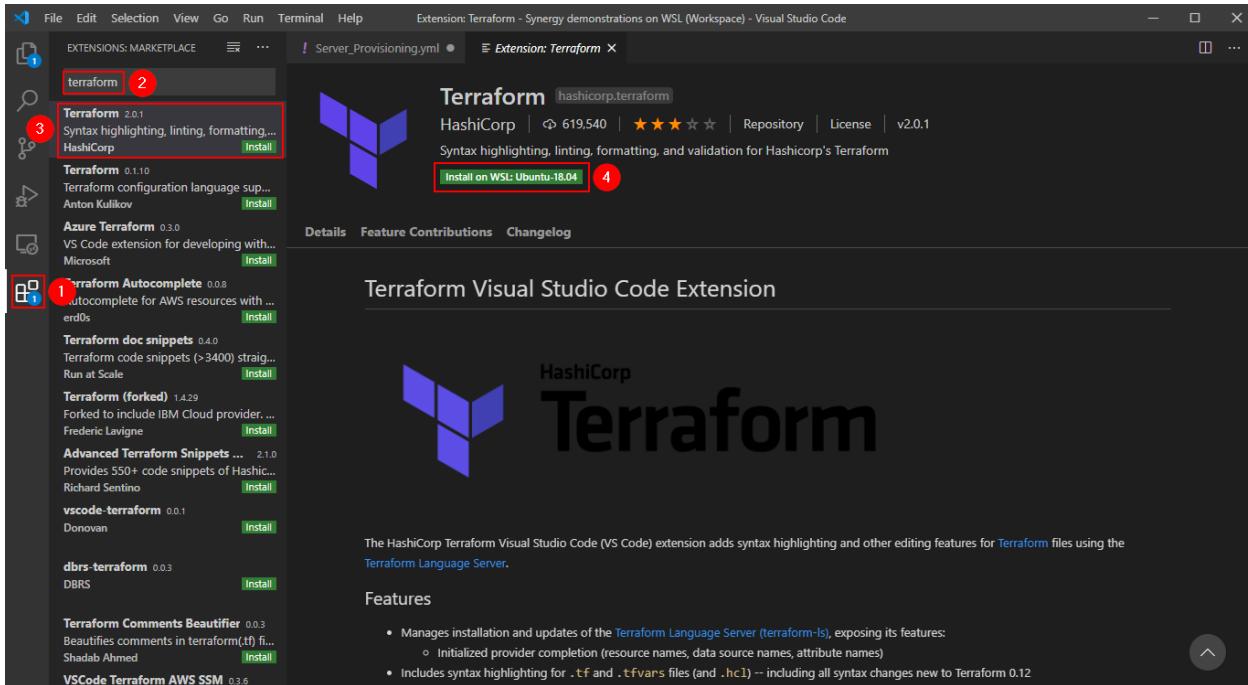
- Once the installation is complete, click on **Reload Required** to activate the extension.



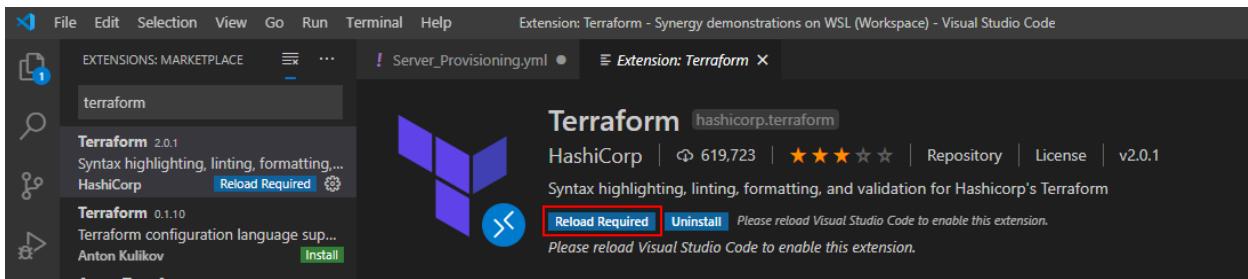
## Preparing VS Code for Terraform

To extend our VS Code editor experience with Terraform, we are going to install a Terraform extension. This extension provides syntax highlighting, linting, formatting and validation. It also supports running Terraform from the WSL Ubuntu instance.

- Go to the **Extensions** and search for the **Terraform** extension from HashiCorp and install it.



- Once the installation is complete, click on **Reload Required** to activate the extension.



## Creating ‘Synergy demonstrations on WSL’ VS Code workspace

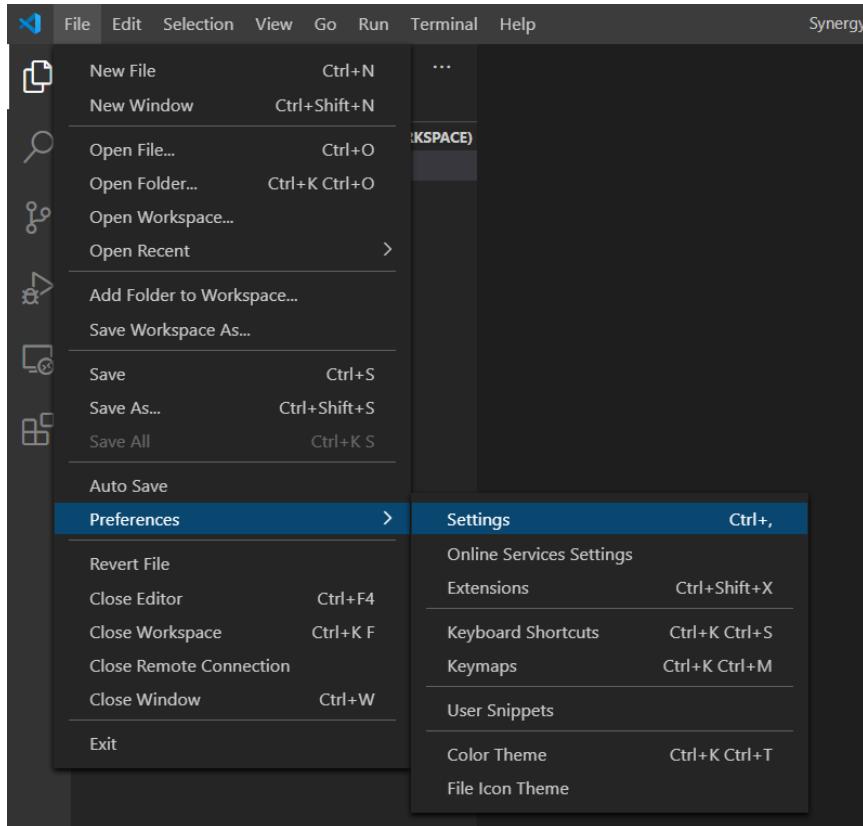
A VS Code workspace is usually just a project that consists of one or more root folders, along with all of the VS Code configurations that belong to that project. These configurations include data such as:

- Settings that should be applied when that project is opened
- Recommended extensions for the project (like the ones we installed earlier)
- Project-specific debugging configurations (Python debugger)

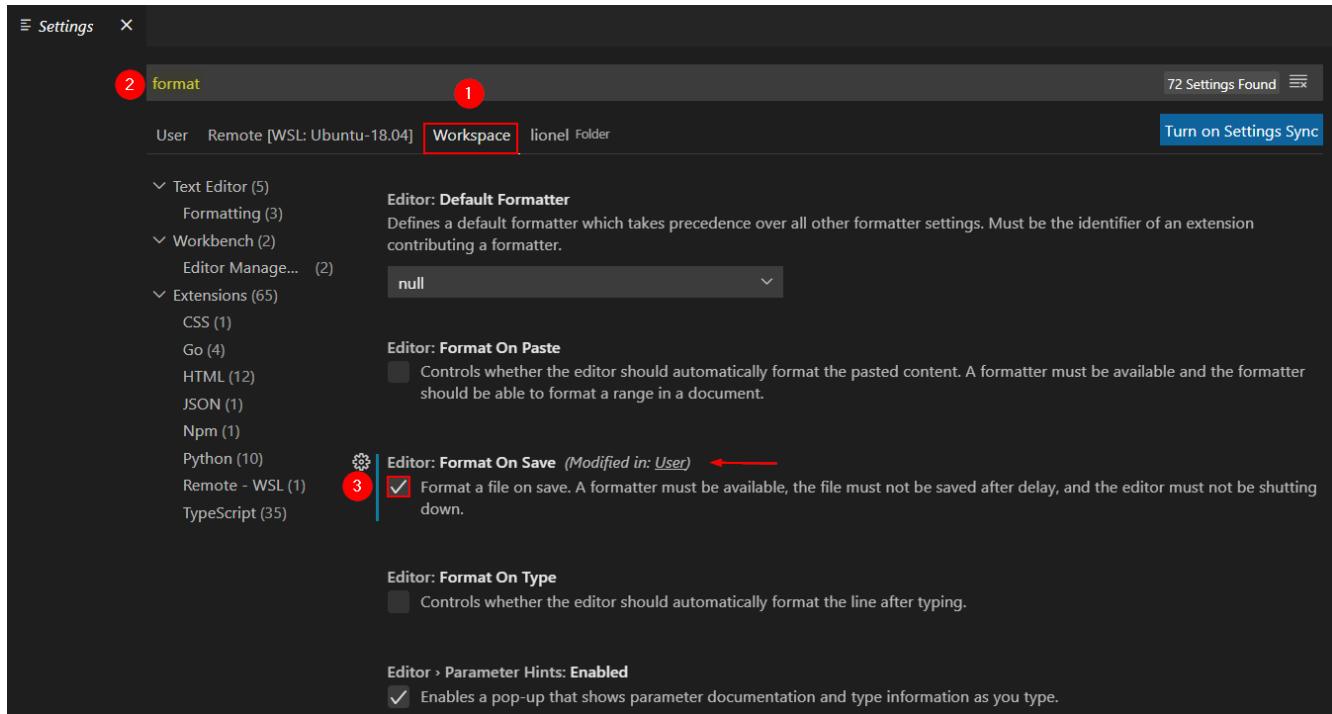
Before saving the workspace, VS Code settings can be changed to meet our needs.

There are tons of VS Code settings that can be changed with some that are very useful. One is really recommended when writing scripts is the **Format On Save**. This “must have” option automatically formats your code when you save your work by improving spacings, indents, by wrapping lines if necessary and fixing quotes.

- To set the VS Code workspace settings, go the **File / Preferences / Settings**



- Select **Workspace** then type in the search field any settings you want to modify, for instance **Format** to enable **Format On Save**



Next important task before saving the workspace, we can clone the **HPE-Synergy-demonstration-environment** repository in our VS Code workspace to get access to all the scenario scripts we are going to use in this guide.

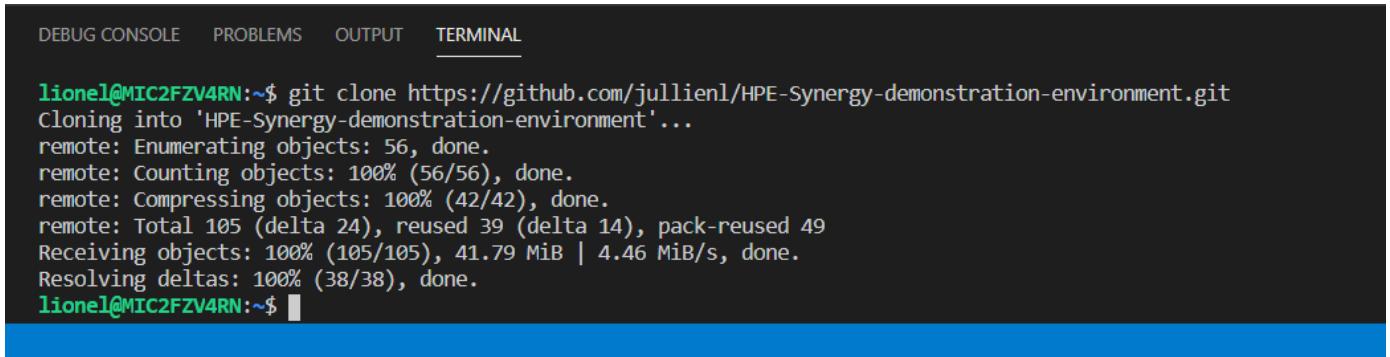
<https://github.com/jullienl/HPE-Synergy-demonstration-environment> repository contains all the Python scripts, Ansible playbook and terraform configuration files that have been developed for the different demonstration scenarios.

- Open the Console and go back to your home directory (/home/<user>):

```
cd ~
```

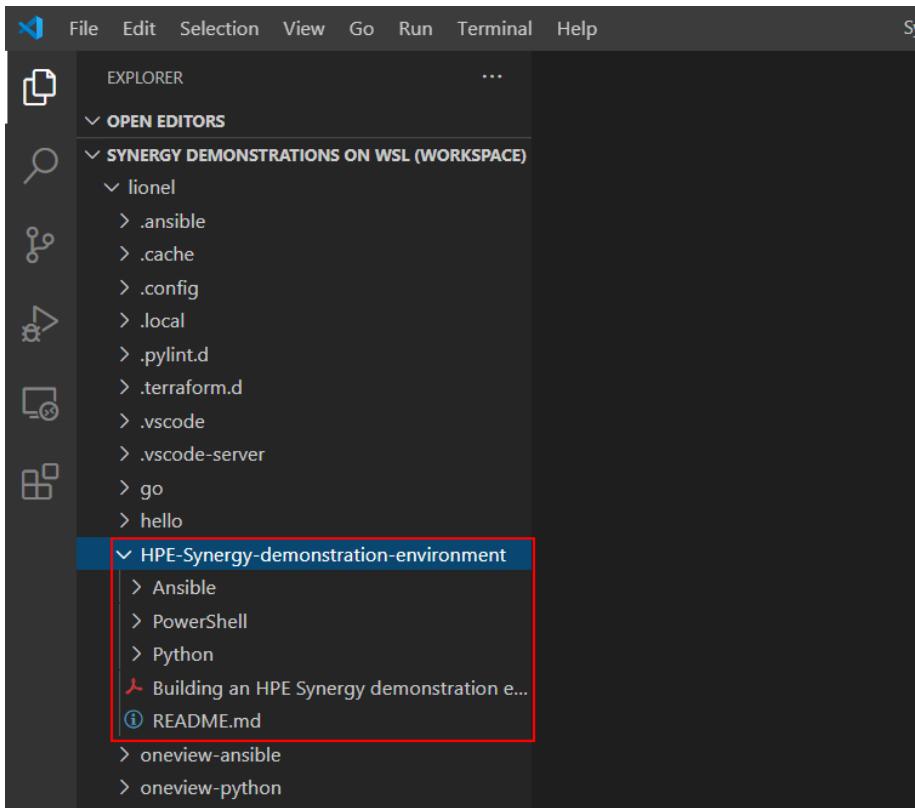
- Then use *git clone* to clone the **HPE-Synergy-demonstration-environment** repository:

```
git clone https://github.com/jullienl/HPE-Synergy-demonstration-environment.git
```



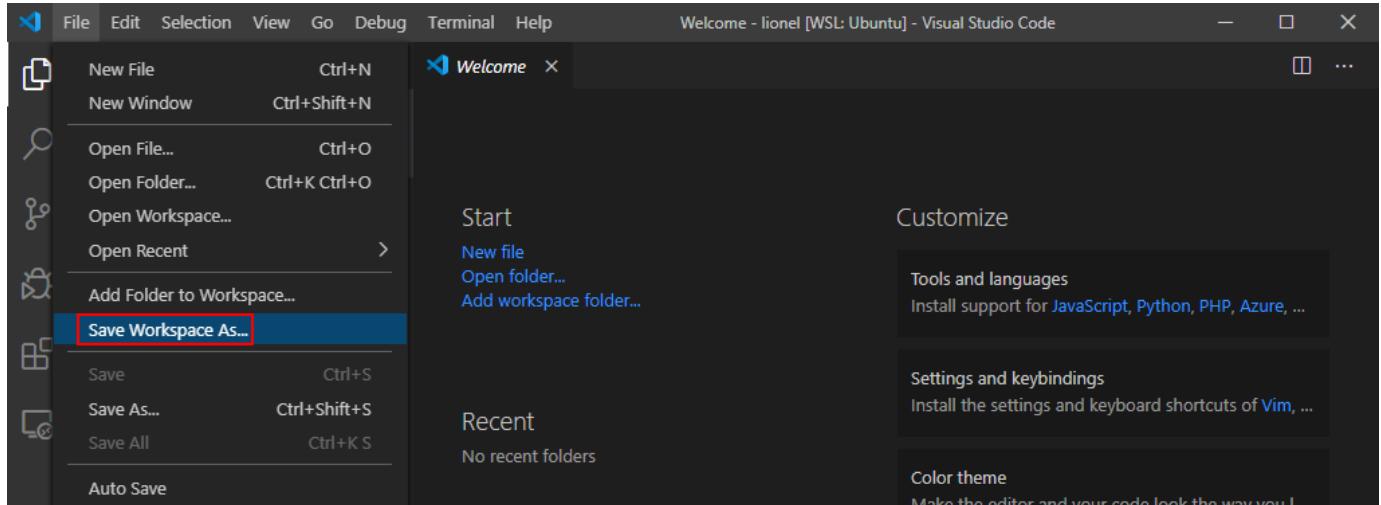
```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
lionel@MIC2FZV4RN:~$ git clone https://github.com/jullienl/HPE-Synergy-demonstration-environment.git
Cloning into 'HPE-Synergy-demonstration-environment'...
remote: Enumerating objects: 56, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 105 (delta 24), reused 39 (delta 14), pack-reused 49
Receiving objects: 100% (105/105), 41.79 MiB | 4.46 MiB/s, done.
Resolving deltas: 100% (38/38), done.
lionel@MIC2FZV4RN:~$
```

- Notice that a new folder **HPE-Synergy-demonstration-environment** is now available in the explorer

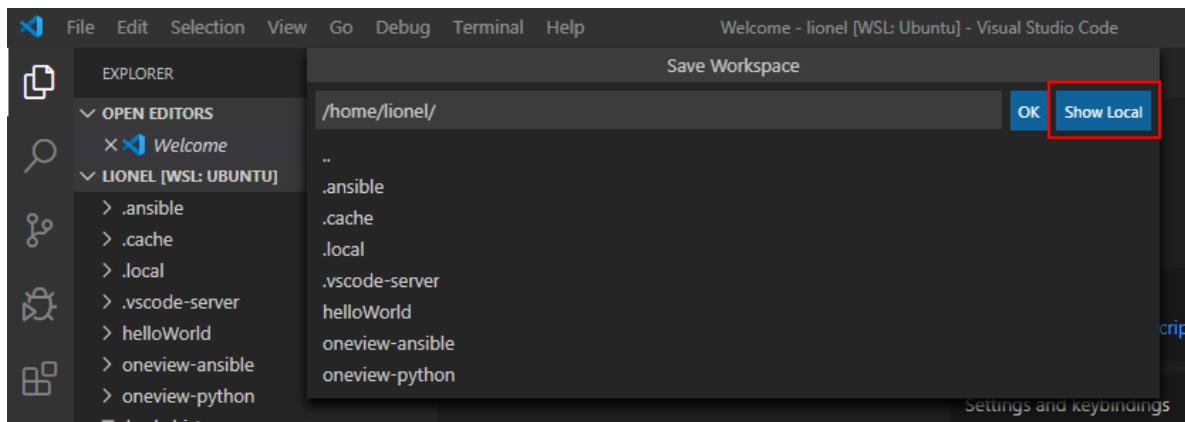


This completes the settings of our WSL workspace, we can now save the current VS Code configuration in a workspace:

- Select **File** menu and click on **Save Workspace as...**:



- Select **Show Local**



- Select a folder (e.g. %HOMEPATH%/documents) and enter a name like **Synergy demonstrations on WSL** then click **Save**

Now each time you open VS Code, you will find the *Synergy demonstrations on WSL* workspace with the Remote – WSL extension opening the Ubuntu folder running in the Windows Subsystem for Linux.

## Preparing VS Code for PowerShell

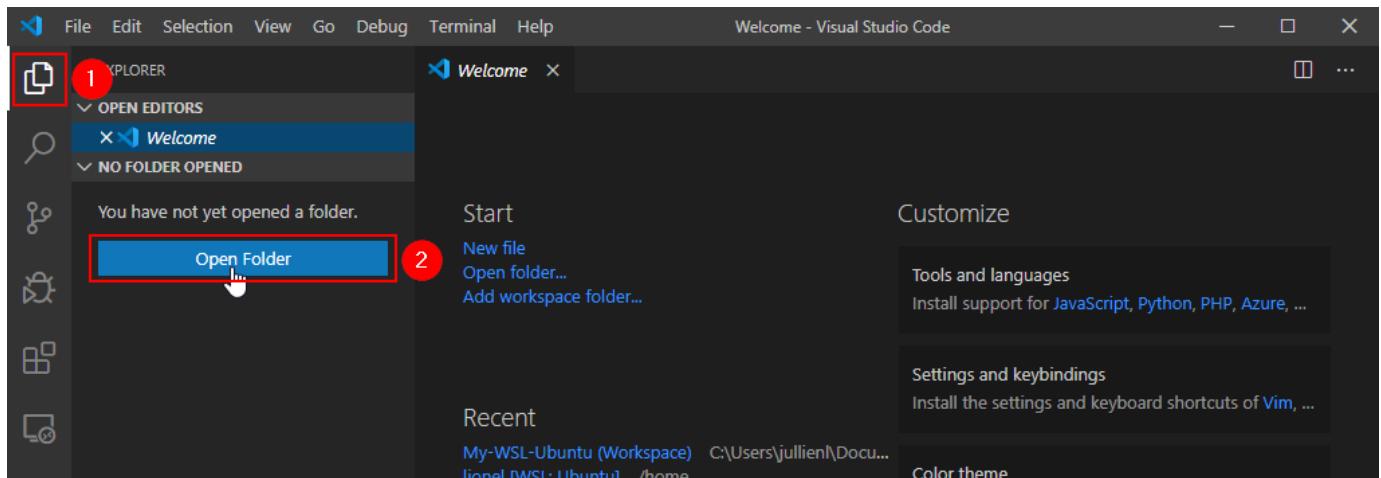
Next, we can install the PowerShell extension to get also nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to cmdlet documentation, ability to run the active script file in the VS Code terminal console, etc.

For PowerShell, we need to use a different VS Code workspace as we cannot mix WSL Linux oriented project with a PowerShell one:

- Open a new VS Code instance by clicking on the Visual Studio Code icon on your desktop:

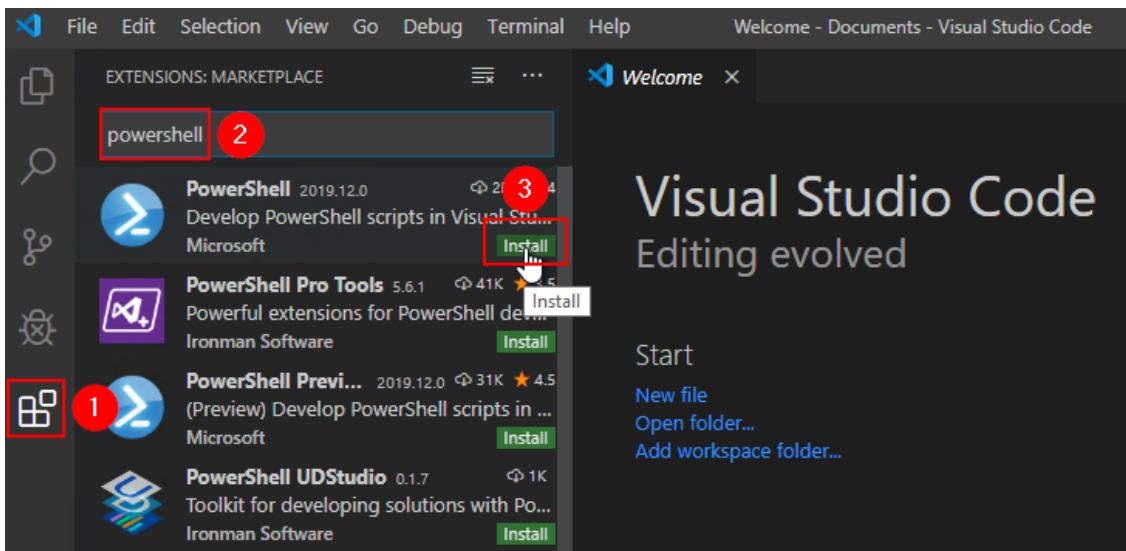


- Then click on the **Explorer** icon on the Activity bar and then click **Open Folder**.

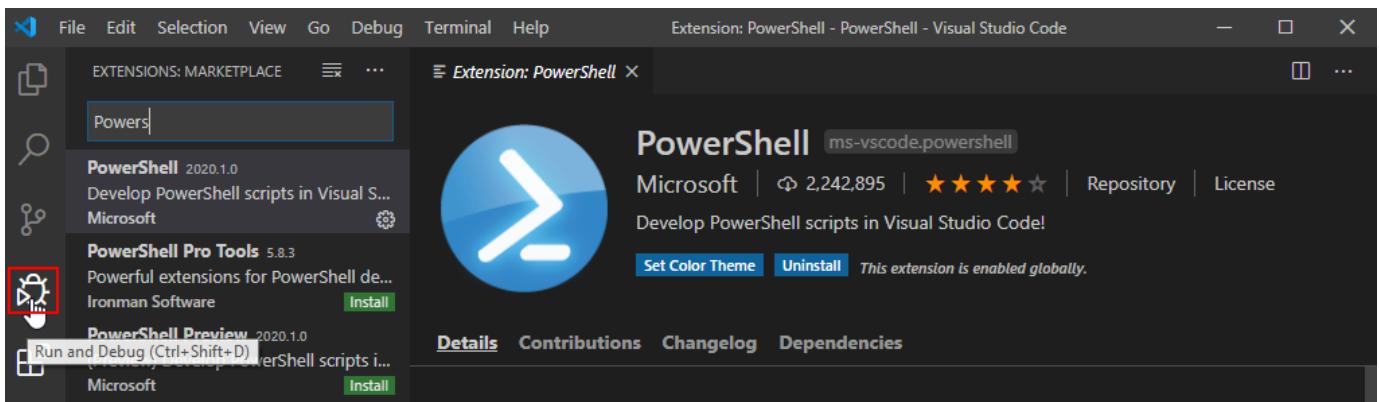


- Open the folder you want your PowerShell project to be in (e.g. \Documents\VS Code projects\PowerShell) and click **Select Folder**.

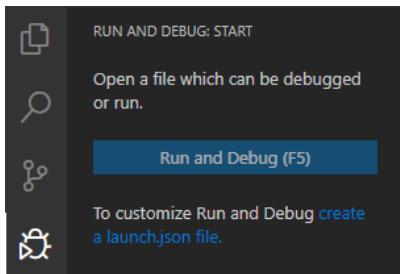
- Go to the **Extensions** pane and search for the **PowerShell** extension and click on the **Install** button:



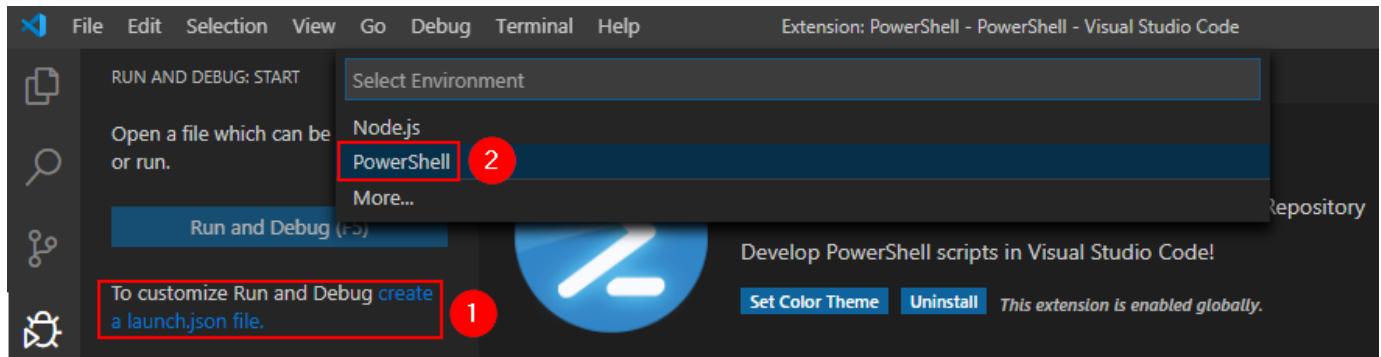
- To configure the PowerShell interpreter once the extension is installed, click on **Run and Debug** on the Activity Bar



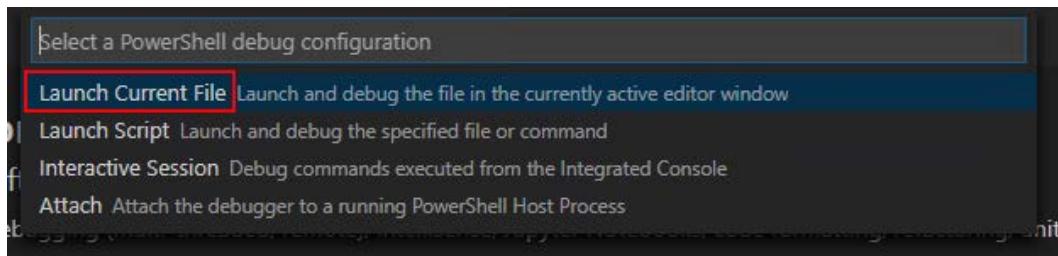
Visual Studio Code's built-in debugger needs some quick setup to run and debug PowerShell scripts. If your system is already configured, you can skip this section. When VS Code is not configured for PowerShell, the following dialog is displayed:



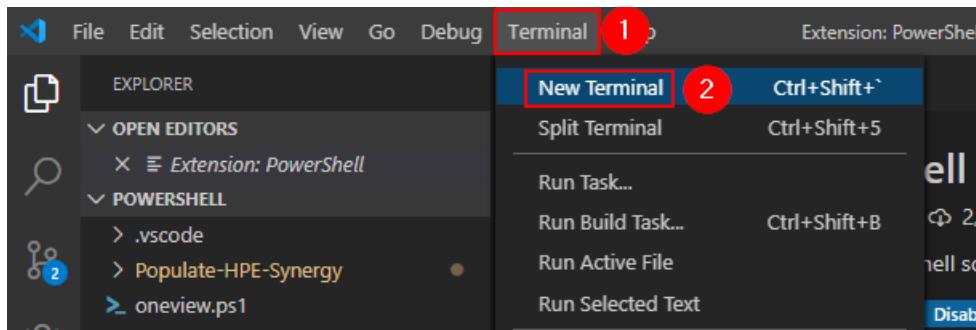
- Click on **Create a launch.json file** to configure Run and Debug, then select **PowerShell**



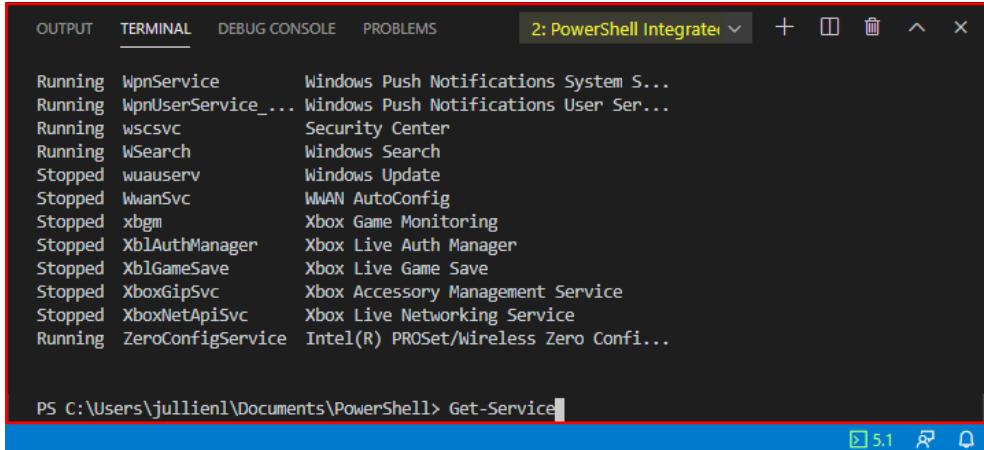
- Then select **Launch Current File**



- Then press **CTRL + S** to save the configuration file then **CTRL + F4** to close the configuration file.
- Select now **Explorer** in the activity bar, a PowerShell Integrated console should start, or you can select **Terminal / New Terminal**



- You can enter **get-service** to make a quick test:



The screenshot shows a PowerShell terminal window titled "2: PowerShell Integrated". The terminal has tabs for "OUTPUT", "TERMINAL", "DEBUG CONSOLE", and "PROBLEMS". The "TERMINAL" tab is active. The command "Get-Service" was run, and its output is displayed. The output lists various Windows services with their status (Running or Stopped) and names. A red box highlights the entire terminal window.

```
PS C:\Users\jullienl\Documents\PowerShell> Get-Service
```

Status	Service Name	Description
Running	WpnService	Windows Push Notifications System S...
Running	WpnUserService_...	Windows Push Notifications User Ser...
Running	wcsvc	Security Center
Running	WSearch	Windows Search
Stopped	wuauserv	Windows Update
Stopped	IwanSvc	WWAN AutoConfig
Stopped	xbgm	Xbox Game Monitoring
Stopped	XblAuthManager	Xbox Live Auth Manager
Stopped	XblGameSave	Xbox Live Game Save
Stopped	XboxGipSvc	Xbox Accessory Management Service
Stopped	XboxNetApiSvc	Xbox Live Networking Service
Running	ZeroConfigService	Intel(R) PROSet/Wireless Zero Config

## Creating ‘Synergy demonstrations on Windows’ VS Code workspace

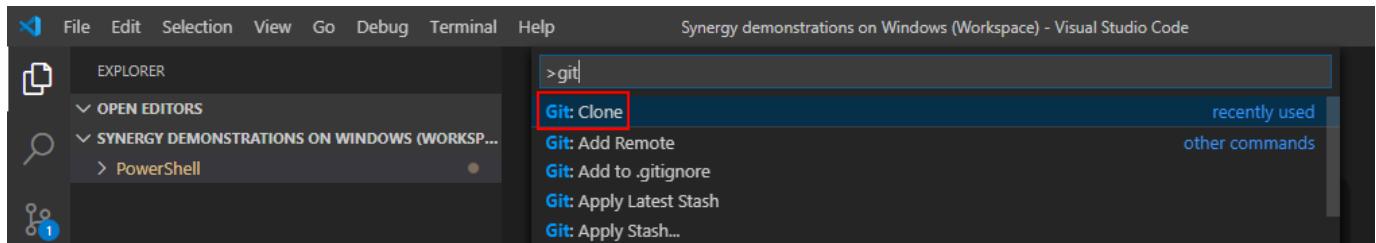
As explained earlier, we cannot mix WSL Linux with Windows/PowerShell projects, so we created a new project that needs to be saved now as our Windows/PowerShell workspace project.

Like with the Linux WSL workspace we created earlier, we can change the current workspace settings to meet our requirements in **File / Preferences / Settings / Workspace**.

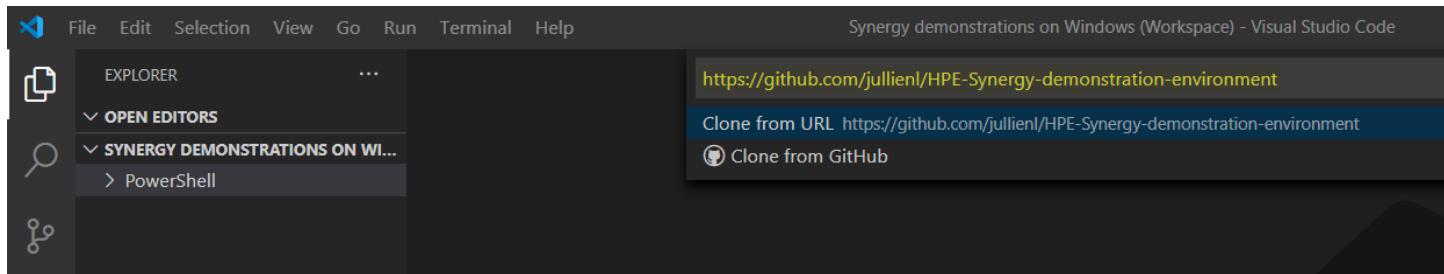
Next, we also need to clone the **HPE-Synergy-demonstration-environment** repository in our VS Code Windows workspace to get access to all the PowerShell scenario scripts we are going to use in this guide.

To clone the repository:

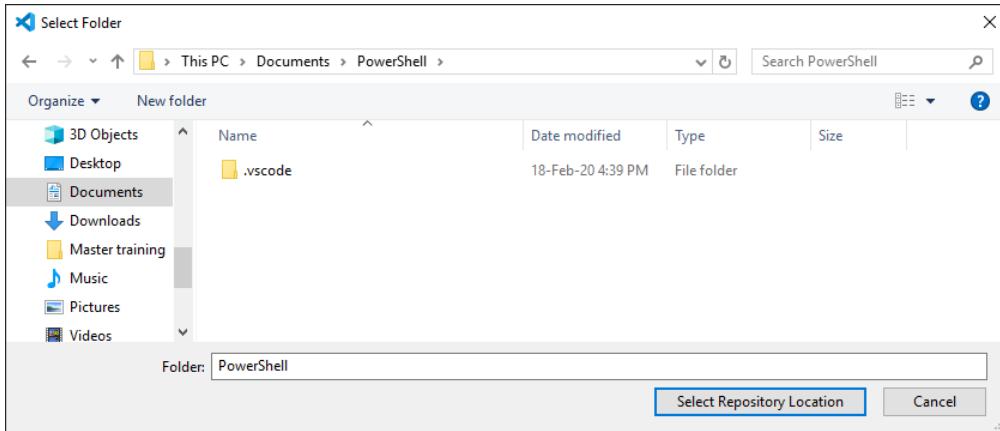
- Open **Synergy demonstrations on Windows** workspace in VS Code.
- Open the Command Palette (**Ctrl+Shift+P**) and enter **Git** and select the **Git: Clone** command:



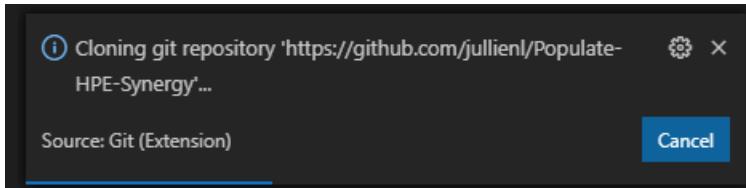
- Enter the following link: <https://github.com/jullienl/HPE-Synergy-demonstration-environment>



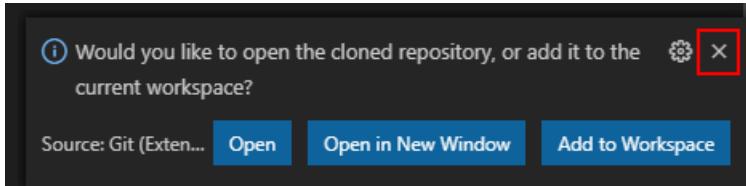
- Then select your Windows user home workspace folder as the parent directory under which to put the repository:



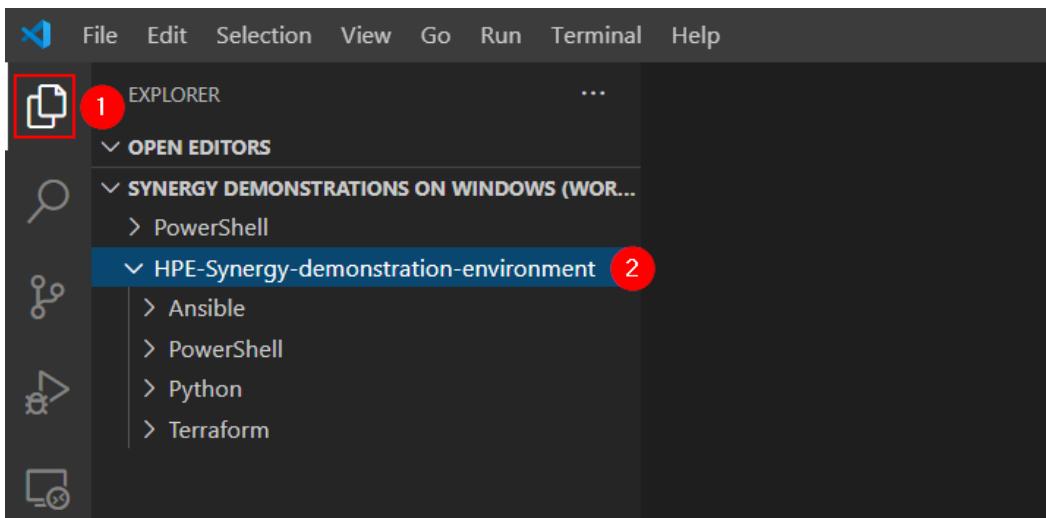
- Once selected, the cloning starts:



- Just close the pop-up window for now



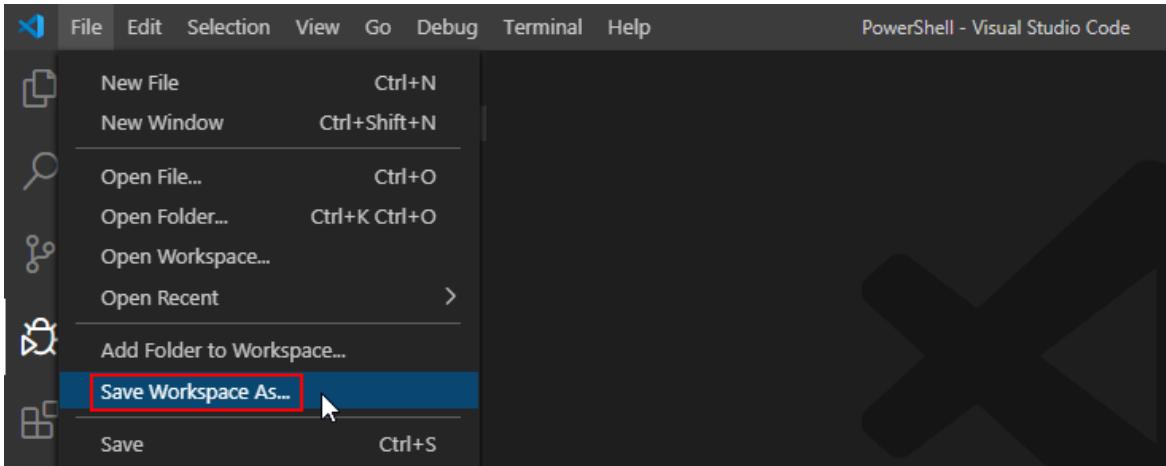
- Notice that a new folder named **HPE-Synergy-demonstration-environment** is now available in the explorer:



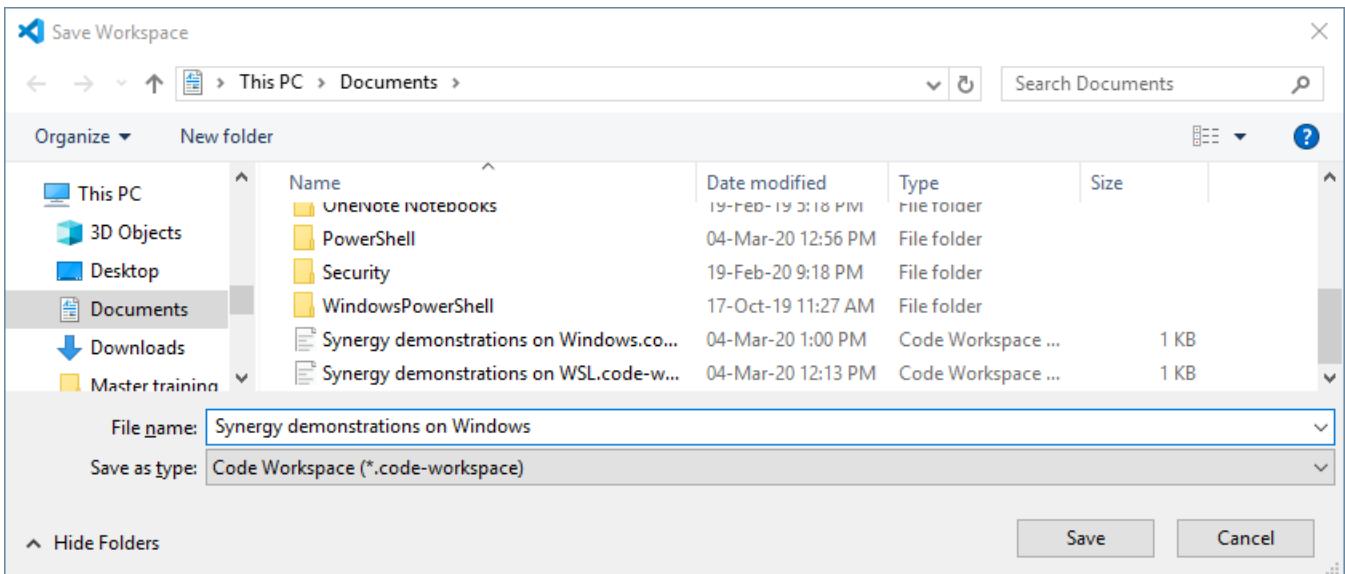
This repository contains all the PowerShell scripts, and other materials used in this guide.

This completes the settings of our Windows workspace, we can now save the current VS Code configuration in a workspace:

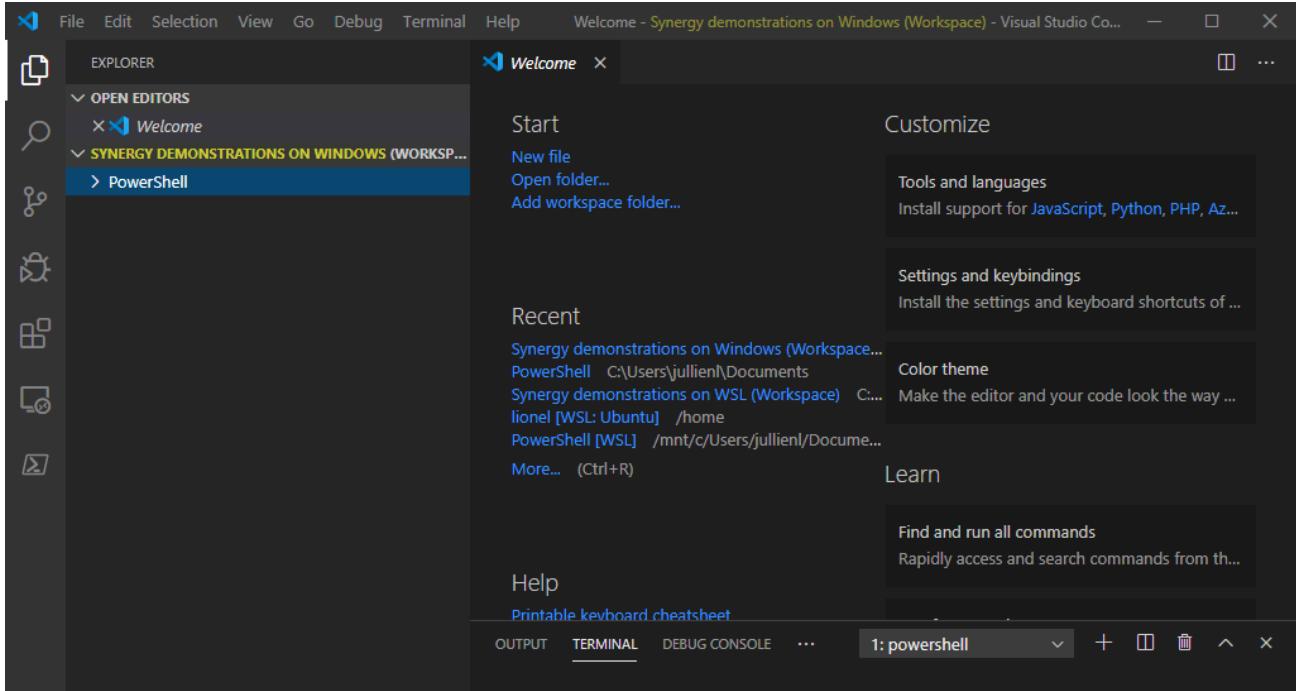
- Select the **File** menu then select **Save Workspace As...**



- Select a folder (e.g. %HOMEPATH%/documents) and enter a name like **Synergy demonstrations on Windows** then click **Save**



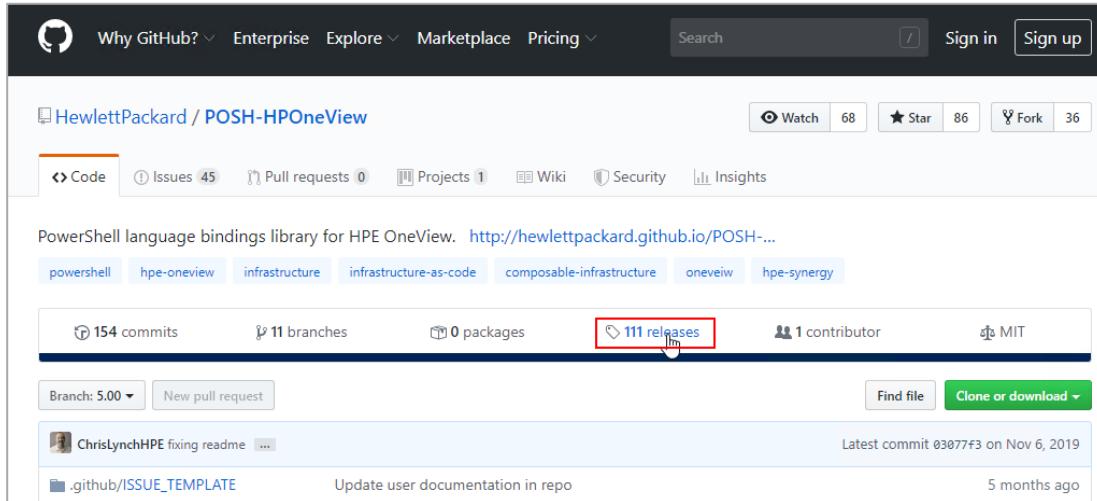
Now like for the WSL workspace, each time you open VS Code, you will find the *Synergy demonstrations on Windows* workspace for the PowerShell activities.



## Installing the PowerShell library for HPE OneView

The PowerShell library for HPE OneView is available on HPE's GitHub site at <https://github.com/HewlettPackard/POSH-HPOneView>, it provides many functions to create nice Composable Infrastructure demonstration for customers.

- Open a browser and visit <https://github.com/HewlettPackard/POSH-HPOneView>
- Select **Releases**



Notice that for 5.x release, we no longer provide any EXE installer so the library can only be installed from the Microsoft PowerShell Gallery.

- Back to Visual Studio Code, enter the following commands in the PowerShell Integrated Console:

```
# Install library from the PowerShell Gallery  
Install-Module HPEOneView.530
```

**Note:** if you need to upgrade from a previous version of the PowerShell library for HPE OneView, refer to the [Upgrading your HPE Synergy demonstration environment chapter](#).

- When prompted, type **Yes** to Install all required providers and **Yes** to install the library from PSGallery

- You can then Import the module using:

```
Import-Module HPEOneView.530
```

- Once the module is successfully imported, you can test the module by entering:

```
get-ovversion
```

```
PS C:\vs Code projects\PowerShell> Get-ovversion

192.168.56.101          LibraryVersion Path
-----
ApplianceVersion: 5.30.00.421400.00 5.30.2507.1303 C:\Program Files\WindowsPowerShell\Modules\HPEOneView.530\5.30.2507.1303

PS C:\vs Code projects\PowerShell>
```

If you get the library version as illustrated above, your module is successfully installed.

**Note:**

If you get the following error:

Import-Module: The library is unable to load due to this system missing the required .Net Framework 4.7.2 client.

You need to install .Net Framework 4.7.2 or later! You can download .Net Framework 4.8 from <https://dotnet.microsoft.com/download/dotnet-framework/net48>. Once installed, retry the import operation.

**Note:**

If you get the following error:

import-module: File <...>.psm1 cannot be loaded because running scripts is disabled on this system

You need to change your system policy to allow scripting. You can enter the following commands:

```
Set-ExecutionPolicy -ExecutionPolicy unrestricted
```

or

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass -Force -Confirm:$False
```

This completes the PowerShell configuration and concludes Chapter-3

In the next chapter, we will configure the demonstration appliance, i.e. HPE OneView.

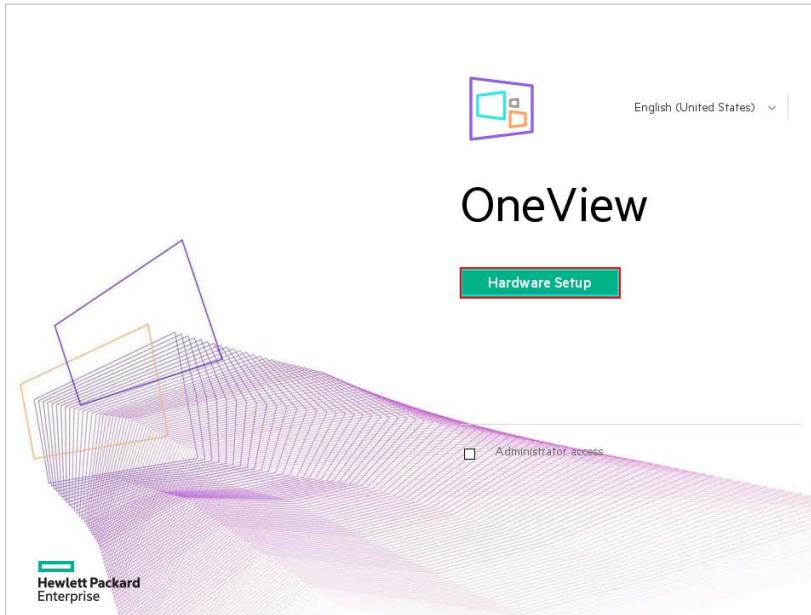


## Chapter-4 -Initial Configuration of the Demonstration Appliance

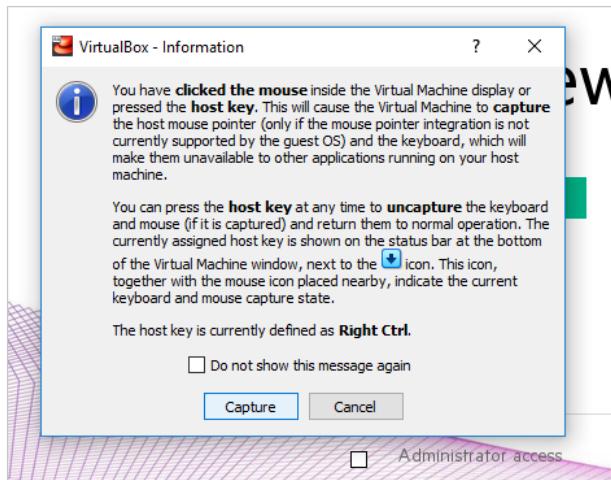
The installation of the scripting languages and the preparation of the Ansible environment is now complete, and OneView has certainly finished booting, we can now finalize the configuration of the appliance.

### Hardware discovery and initial appliance configuration

- Close all open applications running on your computer.
- Important notice:** It is recommended that you close all programs before starting Hardware Setup as the lack of CPU/memory resources during this phase can produce negative effects on the appliance.
- Access to the VirtualBox console and press **Hardware Setup**



**Tip:** You may get a window popping-up, just check **Do not show this message again** and click **Cancel**



- Change the appliance name with **OneView.net**
- Then we can configure three static IP addresses taken from the “Host-only” VirtualBox subnet (192.168.56.0/24)

**Note:** you can open a Windows command prompt and use `Ipconfig` to see the VirtualBox Host-only network adapter subnet

```
C:\Program Files\Oracle\VirtualBox>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

  Connection-specific DNS Suffix  . : lj.lab
  IPv4 Address. . . . . : 192.168.0.25
  Subnet Mask . . . . . : 255.255.252.0
  Default Gateway . . . . . : 192.168.1.1

Ethernet adapter VirtualBox Host-Only Network:

  Connection-specific DNS Suffix  . :
  IPv4 Address. . . . . : 192.168.56.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :
```

- We can enter the following IP addresses for the default “Host-only” VirtualBox subnet:
  - Primary IP address: **192.168.56.101**
  - Subnet mask: **255.255.255.0**
  - Gateway address: **192.168.56.1**
  - Maintenance IP1: **192.168.56.102**
  - Maintenance IP2: **192.168.56.103**



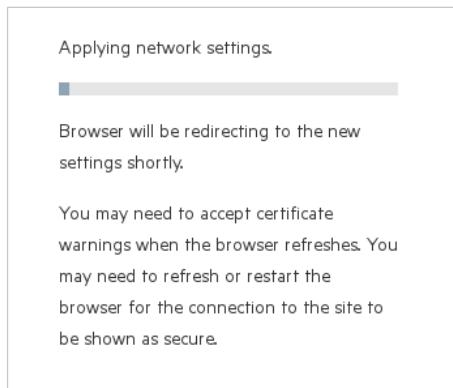
**IPv4**

Address assignment  None  Manual

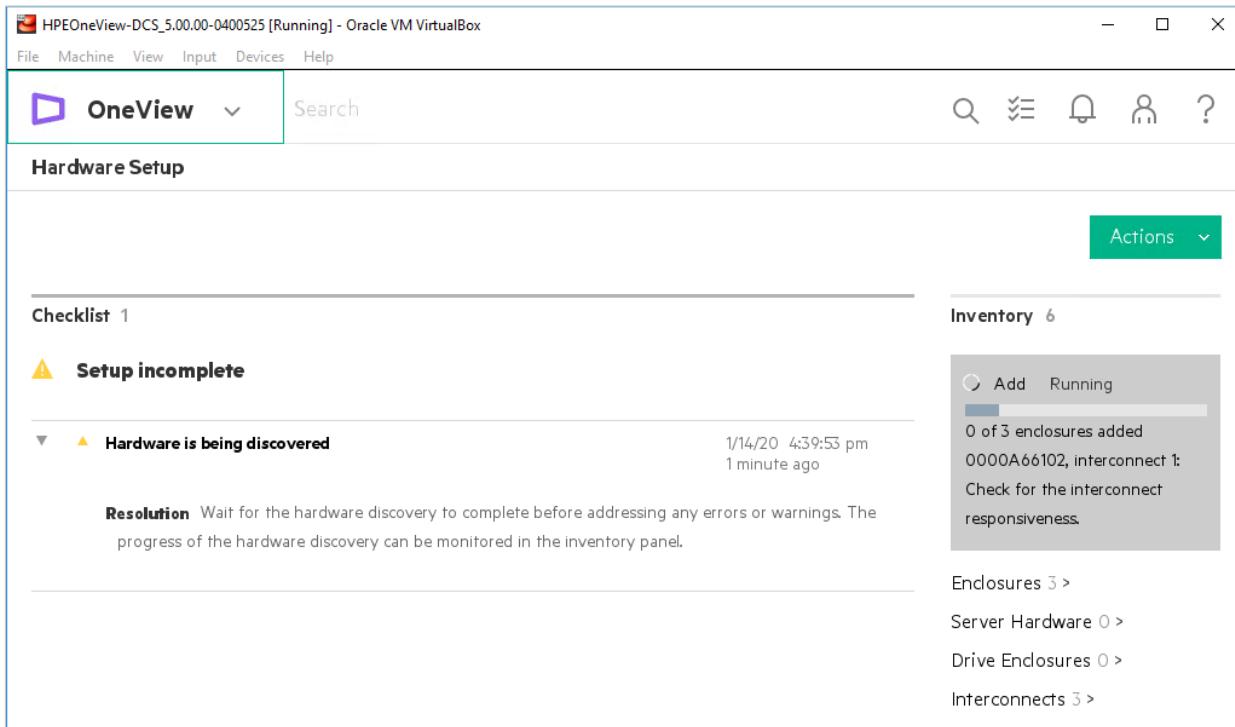
Once IPv4 is disabled, it cannot be re-enabled without reinstalling the appliance. See the OneView install guide for supported configurations when using only IPv6 addresses.  
[Learn more...](#)

IP address	192.168.56.101		
Subnet mask or CIDR	255.255.255.0		
Gateway address	192.168.56.1		
Maintenance IP address 1	192.168.56.102	active	optional
Maintenance IP address 2	192.168.56.103	standby	optional

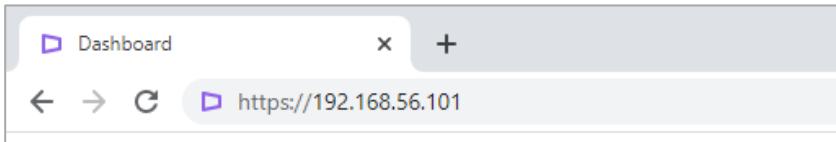
- Then click **OK**



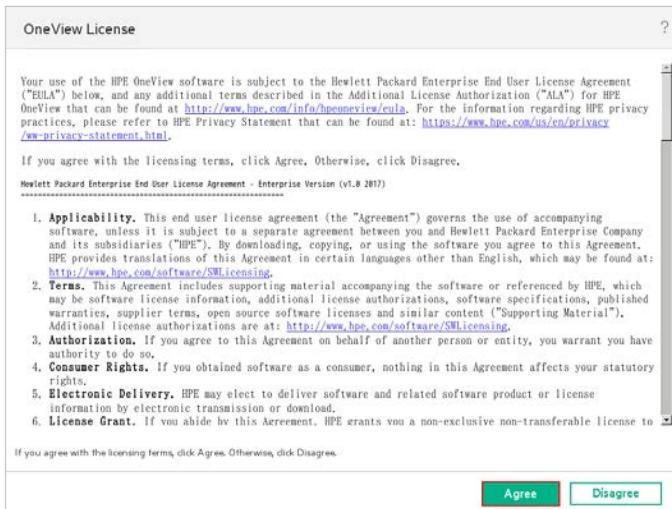
- Once network configuration completes, the initial round of hardware discovery starts:



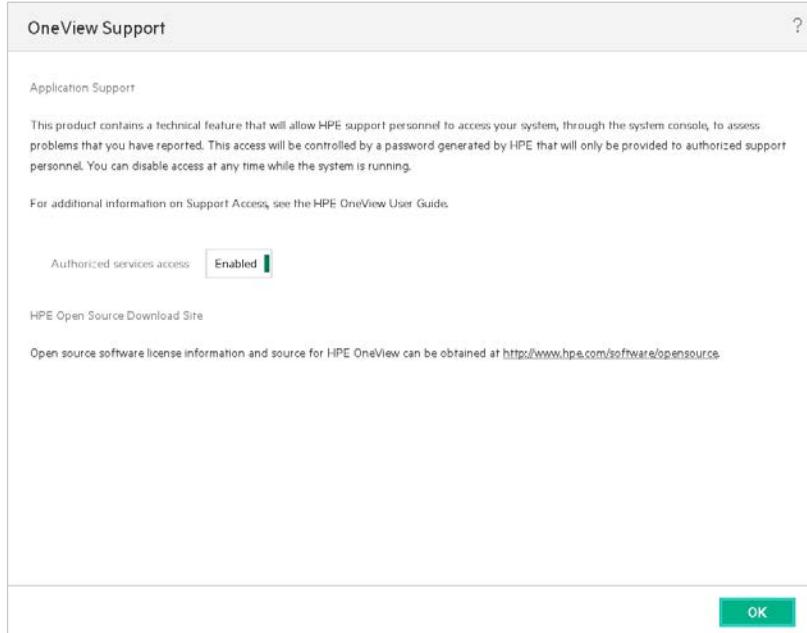
- At this point, you can access your demonstration appliance via a web browser using <https://192.168.56.101>



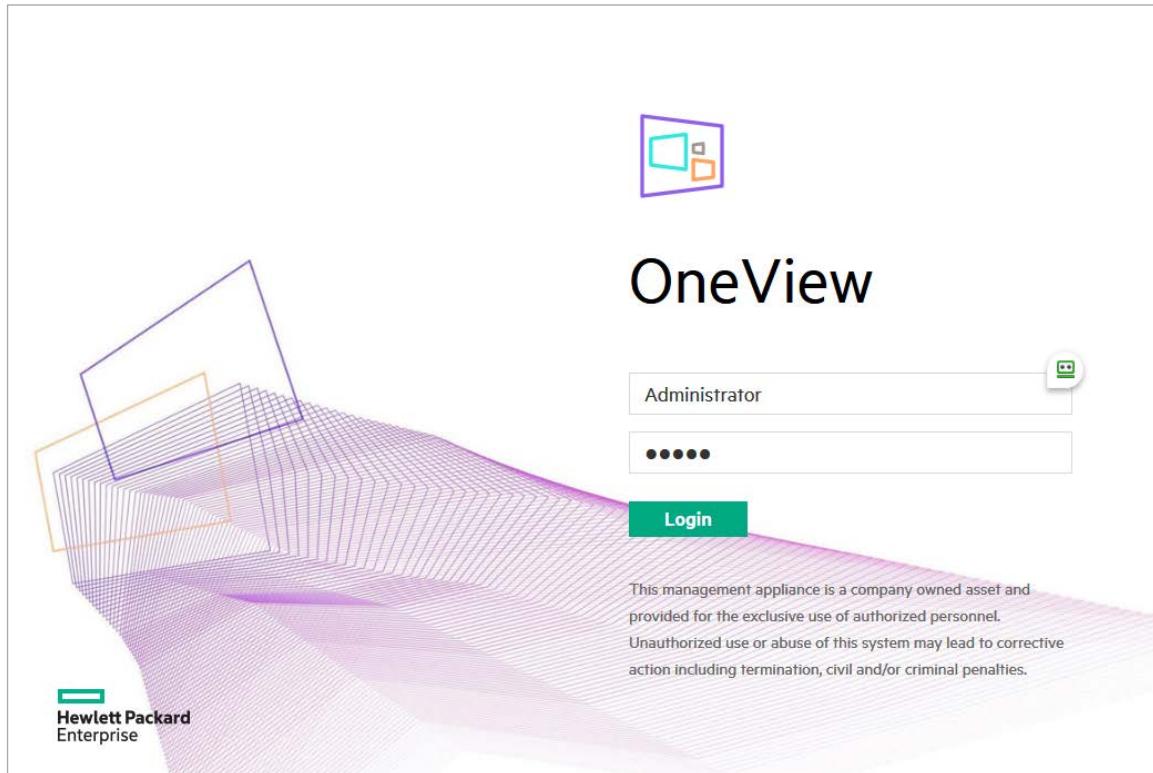
- Accept the OneView license agreement



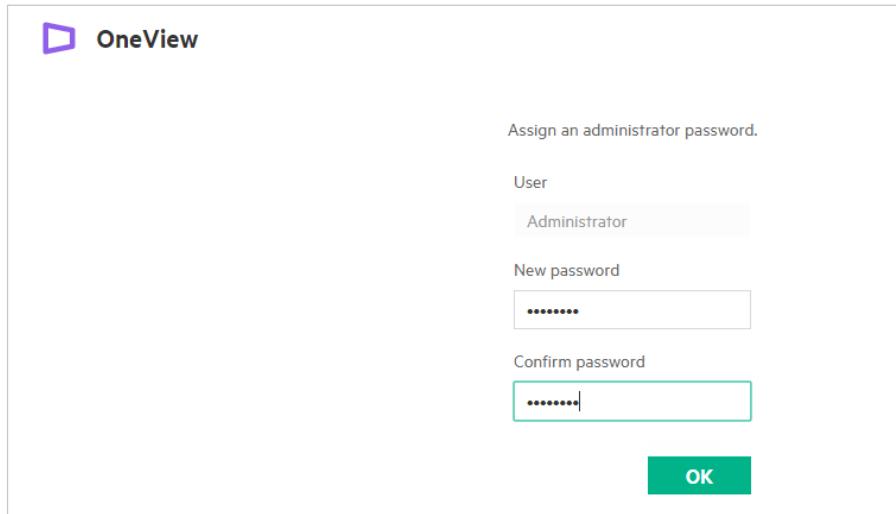
- Leave OneView Support enabled and select **OK**



- Log in using **Administrator / admin**



- Then change the default password to **password**



- The OneView dashboard opens with the tutorial. Click **Close**:

A screenshot of the OneView dashboard. The top navigation bar includes "OneView", "Search", and various icons. The main area is titled "Dashboard" and shows sections for "Server Profiles" (0) and "Server Hardware" (21). A prominent green rectangular overlay box is centered, titled "OneView Tutorial" with the subtext "Welcome to the OneView user interface tutorial. This tutorial will appear the first time you log in from this browser." At the bottom of the overlay are buttons for "< previous", "close" (which is highlighted with a red border), "play", and "next >". Below the overlay, two server status cards are visible: one for "Servers w/ profiles" (21, No profile) and another for "Server Hardware" (21, Not in maintenance).

- Go back to the VirtualBox console to verify the progress of the initial hardware discovery:

The screenshot shows the HPE OneView interface running in Oracle VM VirtualBox. The title bar reads "HPEOneView-DGS\_5.20.00-0419867 (Snapshot 0 - Hardware Setup) [Running] - Oracle VM VirtualBox". The main menu includes File, Machine, View, Input, Devices, and Help. The top navigation bar features the "OneView" logo, a search bar, and various icons for actions like refresh, filter, and help.

**Hardware Setup**

**Checklist 1**

**⚠ Setup incomplete**

**▼ ▲ Hardware is being discovered** 5/27/20 7:50:41 pm  
2 minutes ago

**Resolution** Wait for the hardware discovery to complete before addressing any errors or warnings. The progress of the hardware discovery can be monitored in the inventory panel.

**Inventory 24**

**Add Running**  
0 of 3 enclosures added  
Certificate: Successfully added  
certificate(s):  
fe80:0:0:0:2:0:3:9100eth2

Enclosures 3 >  
Server Hardware 0 >  
Drive Enclosures 3 >  
Interconnects 18 >

- When the initial hardware discovery is completed you should see:

The screenshot shows the HPE OneView interface with the "Actions" button visible. The left sidebar displays the "Inventory 45" section, which includes links to Enclosures (3), Server Hardware (21), Drive Enclosures (3), and Interconnects (18). The main content area lists the following discovered components:

- **3 Enclosures**
- **18 Interconnects**
- **21 Servers**
- **3 Drive Enclosures**

**Important notice:** It is recommended to wait for the discovery to complete as the lack of CPU/memory resources during this phase can produce negative effects on the appliance. The discovery usually takes about 15-20mn.



## Creation of an initial setup snapshot

Once the initial appliance configuration is complete, we can create a first snapshot so that we have the initial configuration saved.

Before creating the snapshot:

- Make sure the initial hardware discovery is complete

The screenshot shows the OneView interface with the title 'OneView' and a search bar. The main area is titled 'Hardware Setup'. On the left, there's a 'Checklist' section with a green checkmark next to 'Hardware discovery complete'. On the right, there's an 'Inventory' section with a count of 45 items, including 'Endlosures 3 >', 'Server Hardware 21 >', 'Drive Enclosures 3 >', and 'Interconnects 18 >'. A green 'Actions' button is located at the top right of the main area.

- It is a good practice to wait 5-6 minutes to give time for the environment to stabilize
- Try to resolve any issues that may be reported by the appliance
- Make sure there is no OneView tasks that is still running
  - Go in OneView / **Activity** then use **Running** in the Filter tool:

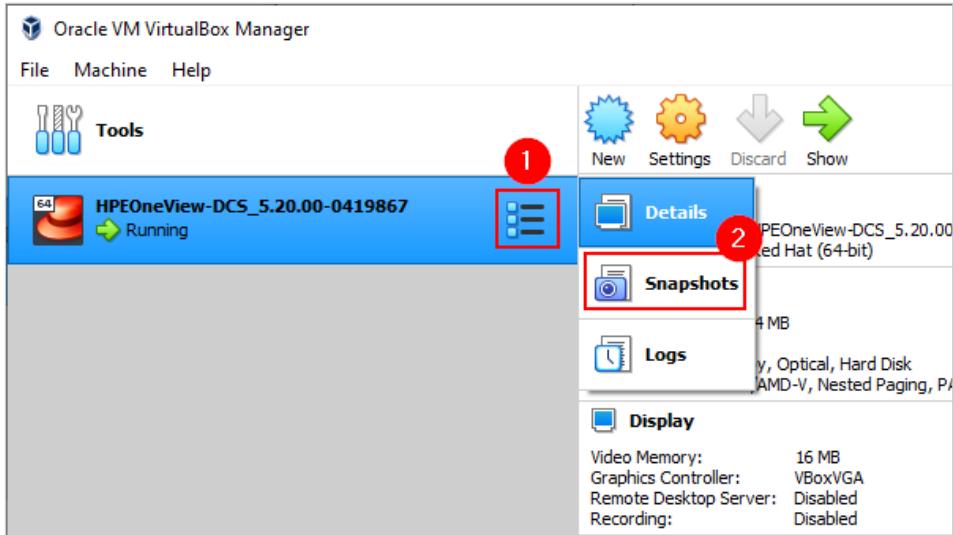
The screenshot shows the OneView 'Activity' page. The search bar contains 'state:running'. The filter tool on the right has a red circle with '1' over it, and the 'Running' option is highlighted with a red border and a red circle with '2'. Other filter options include Pending, Completed, Interrupted, Error, Warning, Suspended, and Cancelling. The main table shows a single row with 'No matches'.

**Note:** Taking a snapshot while the appliance is running is a key practice to get the appliance up and running almost instantaneously.

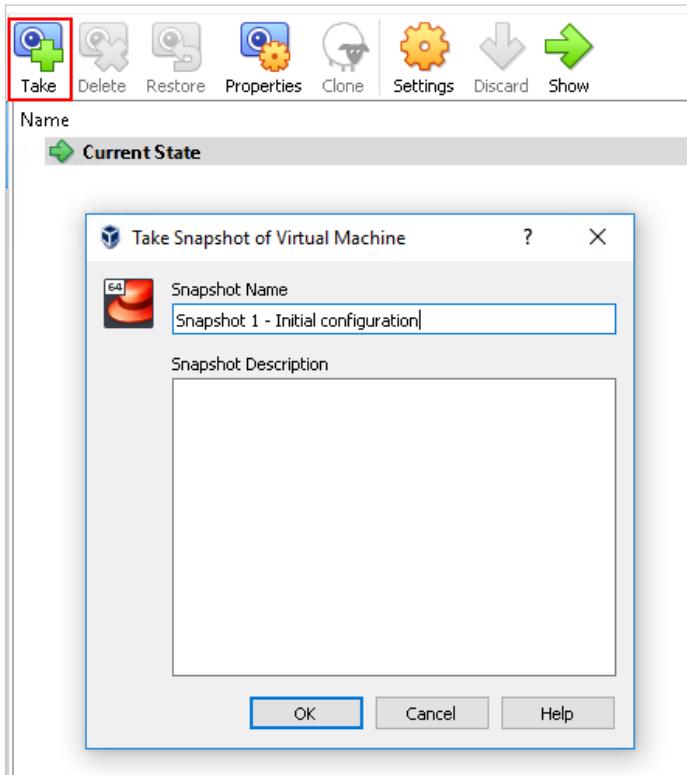


To create a snapshot with the initial configuration:

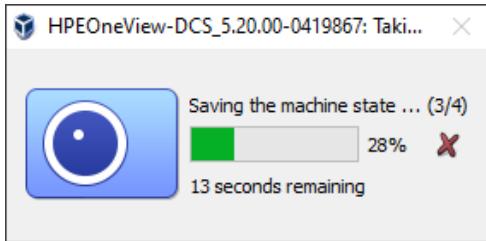
- In VirtualBox, select the **Tools** menu



- Then press on **Take**, enter a snapshot name like **Snapshot 1 – Initial configuration** then click **OK**



- The snapshot creation usually takes about 1 minutes with an SSD drive.



This concludes Chapter-4

In the next chapter, we will install and configure Postman.

## Chapter-5 – Preparing Postman

We have installed VS Code, one of the best editor tools to write and run scripts/playbooks against the appliance, however, if you want to place a REST call to the OneView API without writing any code, we need an additional tool.

Placing a REST call to the OneView API without writing any code can be useful when you want to quickly discover a resource, when you want to identify an API attributes, its components and so on.

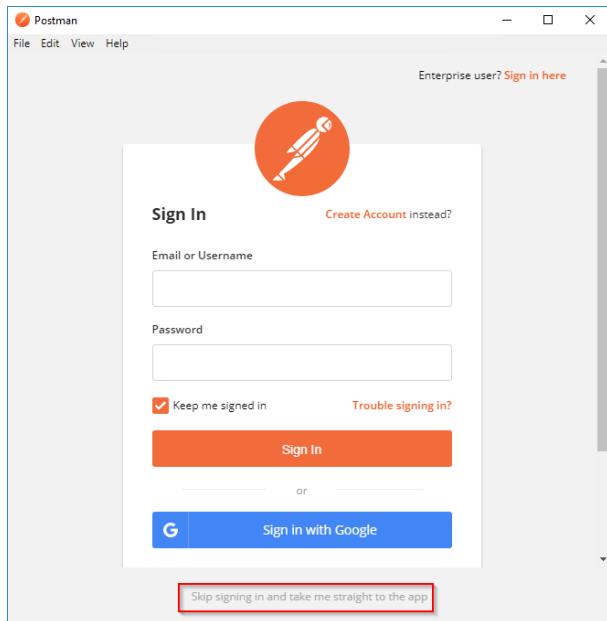
There are several simple solutions for this. My favorite is Postman as it is one of the most complete REST tools and offers useful features:

- You can save your REST calls and share the collections with others
- You can use variables to store for instance the OneView authentication session key

### Installing and configuring Postman

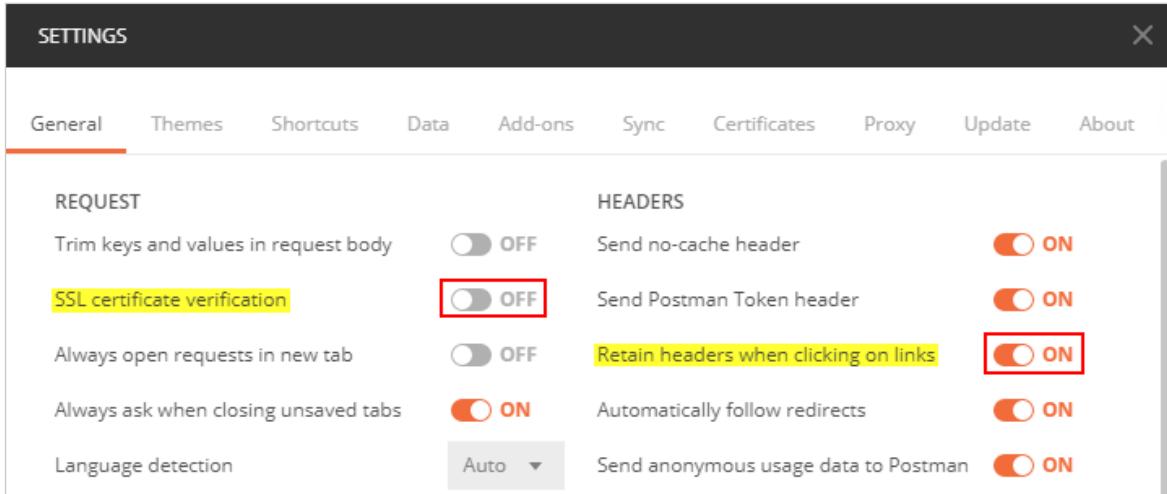
Postman can be downloaded from <https://dl.pstmn.io/download/latest/win?arch=64>

Once installed and started, you may want to create an account if you want to save your work and share your collections with others. Otherwise, you can skip the account creation by clicking on *Skip signing* at the bottom of the page:



With our OneView demonstration appliance, it is necessary to change two default settings.

- Go to **File / Settings** then set **SSL certificate verification** to **OFF** as HPE OneView uses by default a self-signed certificate:



- Set also **Retain headers when clicking on links** to **ON** for greater convenience when clicking on links.

## Importing collections

You can import a OneView API Postman Collection from <https://github.com/jullienl/HPE-Synergy-OneView-demos/tree/master/OneView-Postman-Collections>

This collection brings together several REST calls examples for use with Postman, from the login session to the collection of many different resources using GET requests but also some POST examples to change some settings.

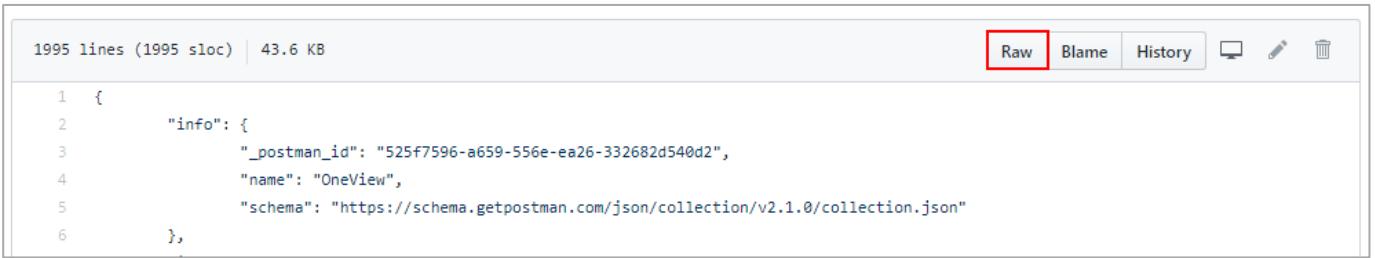
- To import the collection, click on **OneView.postman\_collection.json** to open the file content:

 jullienl New commit	Latest commit 5ef9900 3 days ago
..	
<a href="#">Global Dashboard.postman_collection.json</a>	New commit 3 days ago
<a href="#">OneView.postman_collection.json</a>	New commit 3 days ago
<a href="#">OneView.postman_environment.json</a>	initial commit 9 months ago
<a href="#">README.md</a>	initial commit 9 months ago

**Note:** Right-click then **Save link** as to save the collection on your system will give you a format not recognized error message when importing in Postman.



- Then click **Raw**



1995 lines (1995 sloc) | 43.6 KB

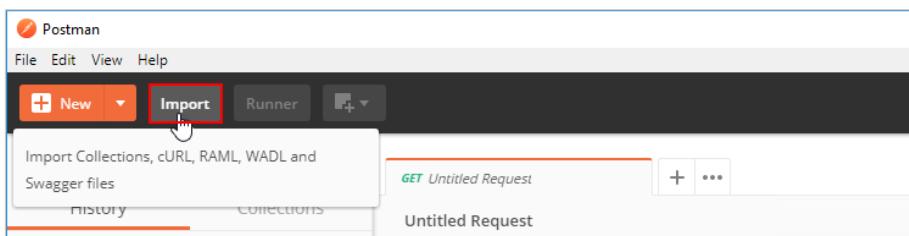
```
1 {
2     "info": {
3         "_postman_id": "525f7596-a659-556e-ea26-332682d540d2",
4         "name": "OneView",
5         "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
6     },
}
```

Raw Blame History

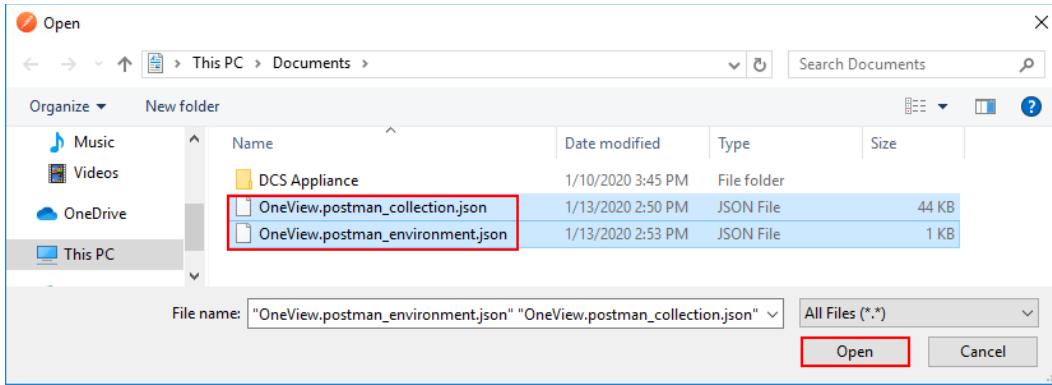
- Then right-click the page and click on **Save as...** to save the JSON file on your system:



- Do the same for **OneView.postman\_environment.json**.
- Back in Postman, click on **Import**:



- Then select the two files and click **Open**



- You should see in collections, the new OneView collection

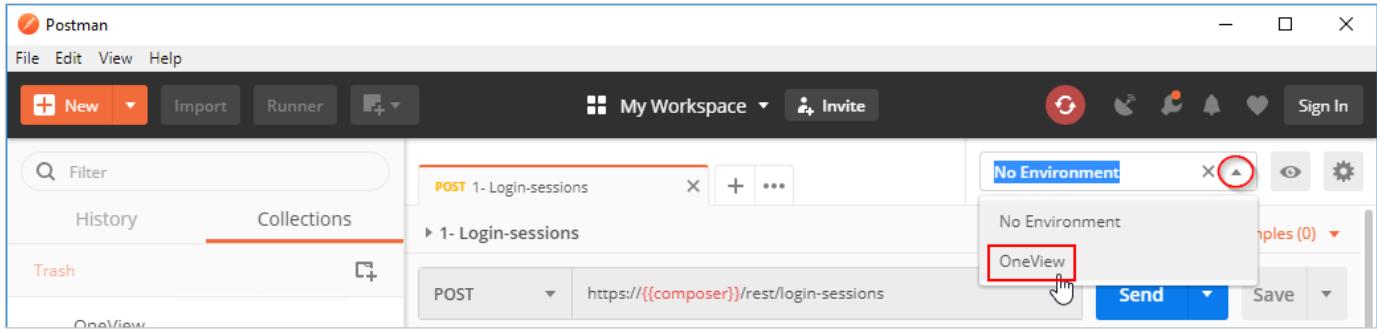
A screenshot of the Postman application interface. On the left, the sidebar shows 'History', 'Collections' (highlighted with a red box), 'Trash', and a folder named 'OneView' containing 49 requests. The main panel displays a 'GET Untitled Request' with the method set to 'GET'. Under the 'Params' tab, there is a 'KEY' section with a single entry 'Key'. Below it, the 'Response' tab is visible. A list of API endpoints is shown on the left side of the main panel:

- POST 1- Login-sessions
- GET 2- Get-X-API-Version
- GET Profile Templates
- GET Get-Ethernet-Networks
- GET Get-Datacenters
- GET Get the OS Deployment plans from ...
- GET Get the Hypervisor Managers from ...
- GET Get the Hypervisor Cluster Profiles ...
- PUT Upload CRL
- GET Get-Interconnect

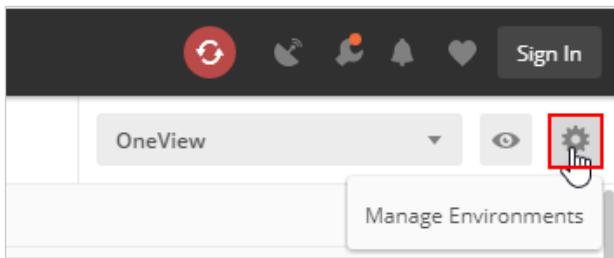
## Configuration of the Postman environment

This collection works with a Postman environment (the second file that was imported) which provide a set of variables that allow us to reuse header values in different REST requests. This really simplifies the request creation and the overall use of Postman.

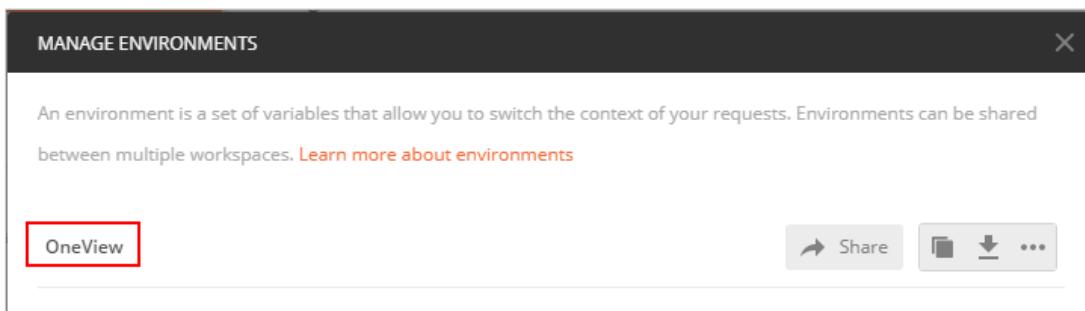
- Expand the environment menu and select **OneView**



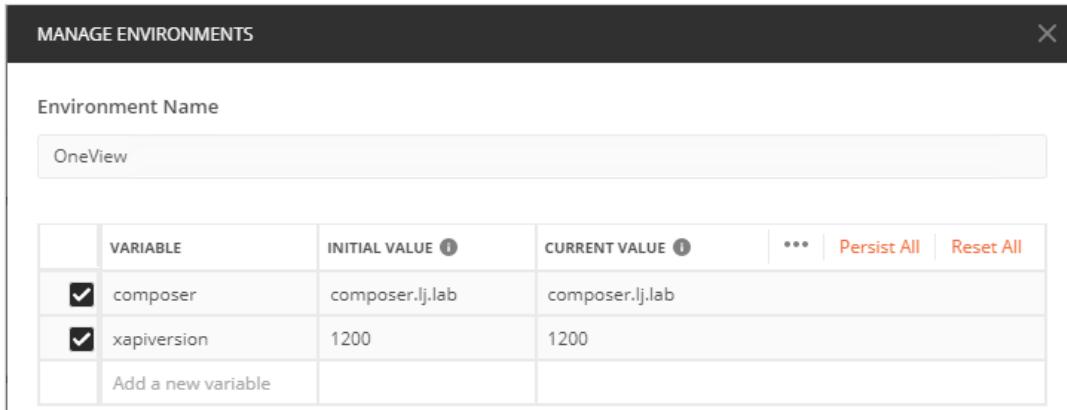
- Then select **Manage Environments**



- Click on **OneView**



There are two variables defined as illustrated below:

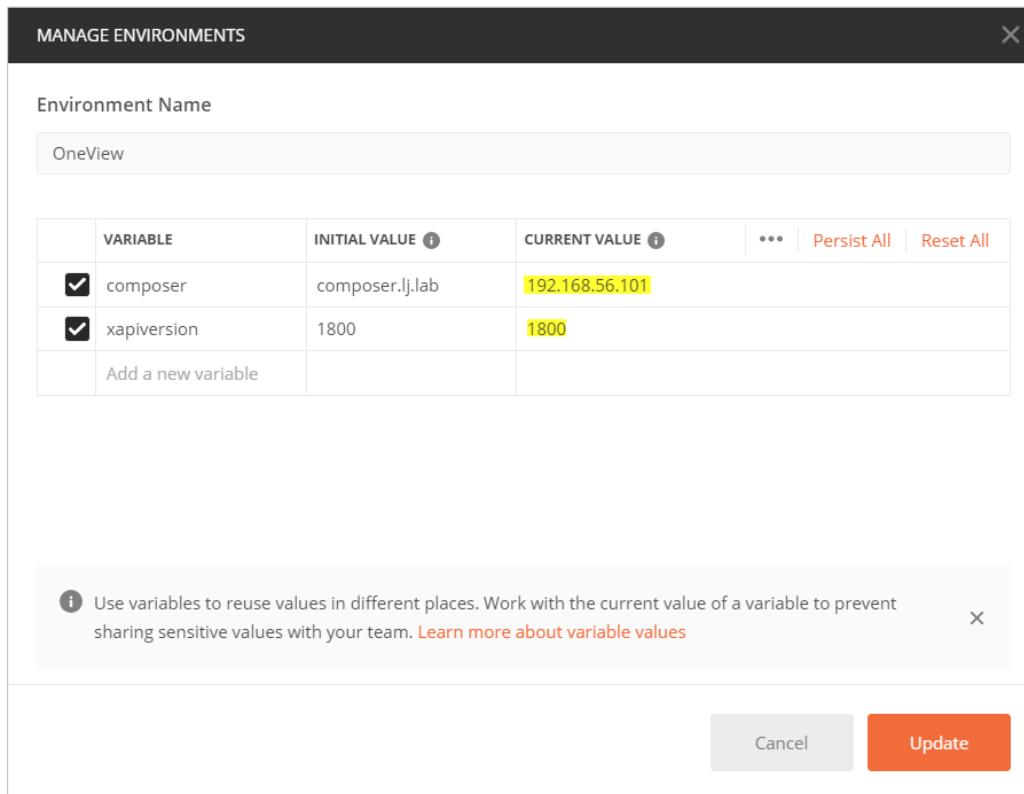


The screenshot shows the 'MANAGE ENVIRONMENTS' dialog box. At the top, it says 'Environment Name' and has a field containing 'OneView'. Below this is a table with three columns: 'VARIABLE', 'INITIAL VALUE', and 'CURRENT VALUE'. There are two rows in the table. The first row has a checked checkbox, the variable 'composer', the initial value 'composer.lj.lab', and the current value 'composer.lj.lab'. The second row also has a checked checkbox, the variable 'xapiversion', the initial value '1200', and the current value '1200'. At the bottom right of the table are buttons for 'Persist All' and 'Reset All'.

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
<input checked="" type="checkbox"/> composer	composer.lj.lab	composer.lj.lab
<input checked="" type="checkbox"/> xapiversion	1200	1200
Add a new variable		

We need to modify the *composer* and *xapiversion* variables to match with our configuration.

- Change the *composer* value with **192.168.56.101**, the IP address of the appliance then press **Update**
- Change the *xapiversion* value with **1800**, the API version of OneView 5.30 then press **Update** then **Close**



The screenshot shows the 'MANAGE ENVIRONMENTS' dialog box with the same structure as before. The 'composer' variable now has a yellow background in the 'CURRENT VALUE' column, indicating it has been modified. The 'xapiversion' variable also has a yellow background in its 'CURRENT VALUE' column. A tooltip message at the bottom left says: 'Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#)'.

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
<input checked="" type="checkbox"/> composer	composer.lj.lab	192.168.56.101
<input checked="" type="checkbox"/> xapiversion	1800	1800
Add a new variable		

At the bottom right, there are 'Cancel' and 'Update' buttons. The 'Update' button is highlighted with a red box.

- To test a POST /REST/login-sessions with the demonstration appliance, select **1-Login-sessions**

The screenshot shows the Postman application interface. At the top, there are buttons for '+ New', 'Import', 'Runner', and 'My Workspace'. Below that is a search bar labeled 'Filter' and a navigation bar with 'History', 'Collections' (which is highlighted in orange), and 'Trash'. On the left, a sidebar shows a tree structure with 'OneView' expanded, containing 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. Under 'OneView', there are 49 requests. A red box highlights the 'POST 1- Login-sessions' item. The main workspace shows a POST request to 'https://{{composer}}/rest/login-sessions'. The 'Params' tab is selected, showing a table with columns 'KEY', 'VALUE', 'DESCRIPTION', and '\*\*\* Bulk Edit'. There is one row with 'Key' and 'Value' fields. The 'Headers' tab shows '(2)' headers. The 'Body' tab is selected and has a green dot, indicating it's active. The 'Tests' tab has a green dot. Other tabs include 'Cookies', 'Code', and 'Comments (0)'. At the bottom right are 'Send' and 'Save' buttons.

- Then we need to modify the password used by Administrator, select **Body** then enter the password you set in the previous section then click **Save**

This screenshot shows the same Postman interface as above, but with modifications. The 'Body' tab is now selected, indicated by a red box and a red circle with the number '1'. In the body editor, there is a JSON payload entered:

```

1
2 {  

3   "authLoginDomain": "Local",  

4   "password": "password",  

5   "userName": "administrator"  

6 }

```

Below the body editor, there are several radio buttons for selecting the content type: 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected and highlighted in blue), and 'binary'. To the right of these buttons is a dropdown menu set to 'JSON (application/json)'. The 'Save' button at the top right is also highlighted with a red box and a red circle with the number '3'.

- Then press **Send**

The response we should get is the following:

Body Cookies Headers (12) Test Results Status: 200 OK Time: 86 ms Size: 428 B Save Download

Pretty Raw Preview JSON

```

1 <pre>{
2   "sessionID": "LTiyNjY0MTk3Mzkxz2a1n_AKtwh9jH1PSVOnW3EB4EQ0qnkT",
3   "partnerData": {}
4 }</pre>

```

In the Response section, as illustrated above, you want to check the HTTP status code. A value of 200 means it was successful. In the body section, you should get a session ID for the authentication.

All other REST requests available in this OneView collection should work successfully as we are passing, using a variable, this session ID to all requests. The creation of this variable `sessionID` is done in the **Tests** menu of the Login-sessions request:

1- Login-sessions Examples (0) ▾

POST https://{{composer}}/rest/login-sessions Send Save

Params Authorization Headers (2) Body Pre-request Script Tests (0) Cookies Code Comments (0)

```

1 var jsonData = JSON.parse(responseBody);
2 postman.setEnvironmentVariable("sessionID", jsonData.sessionID
    );

```

Test scripts are written in JavaScript, and are run after the response is received.

The variable `sessionID` is then set using `{{...}}` in the header of each request:

History Collections Examples (0) ▾

OneView 49 requests

- Logs
- Reserved VLAN range
- eFuse
- Associations
- 1- Login-sessions
- 2- Get-X-API-Version
- Profile Templates

2- Get-X-API-Version Examples (0) ▾

GET https://{{composer}}/rest/version Send Save

Params Authorization Headers (2) Body Pre-request Script Tests (0) Cookies Code Comments (0)

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Prese
<input checked="" type="checkbox"/>	X-API-Version	300				
<input checked="" type="checkbox"/>	Auth	<code>{{sessionID}}</code>				
	Key	Value	Description			

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK Time: 40 ms Size: 391 B Save Download

Pretty Raw Preview JSON

This concludes Chapter-5

In the next chapter, we will prepare a demo scenario.

## Chapter-6 - Preparing your demonstration appliance for demos

HPE Synergy is a new class of system that falls under a category known as a composable infrastructure. This category is emerging as the datacenter infrastructure that seeks to reconstruct previously dedicated compute, storage and network fabric resources into shared, flexible resource pools that are available for on-demand allocation.

This on-demand availability and flexibility of all resources, identified as the core of the new composable infrastructure can be effectively demonstrated using scripting languages such as PowerShell, Python and others.

In this chapter, we are going to prepare the appliance to be ready for customer facing demonstrations.

If we need a fast method to run any type of demonstration, we must use snapshot technology and prepare at least 2 snapshots for different type of scenarios/use cases:

- 1- First snapshot: OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network)
  - ⇒ Can be used to show how to automate the setup of Synergy and the power of our infrastructure as code implementation.
- 2- Second snapshot: same as first snapshot but here Synergy frames are fully configured with LE, EG, LIG and some networks.
  - ⇒ Can be used to run demos with an already configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

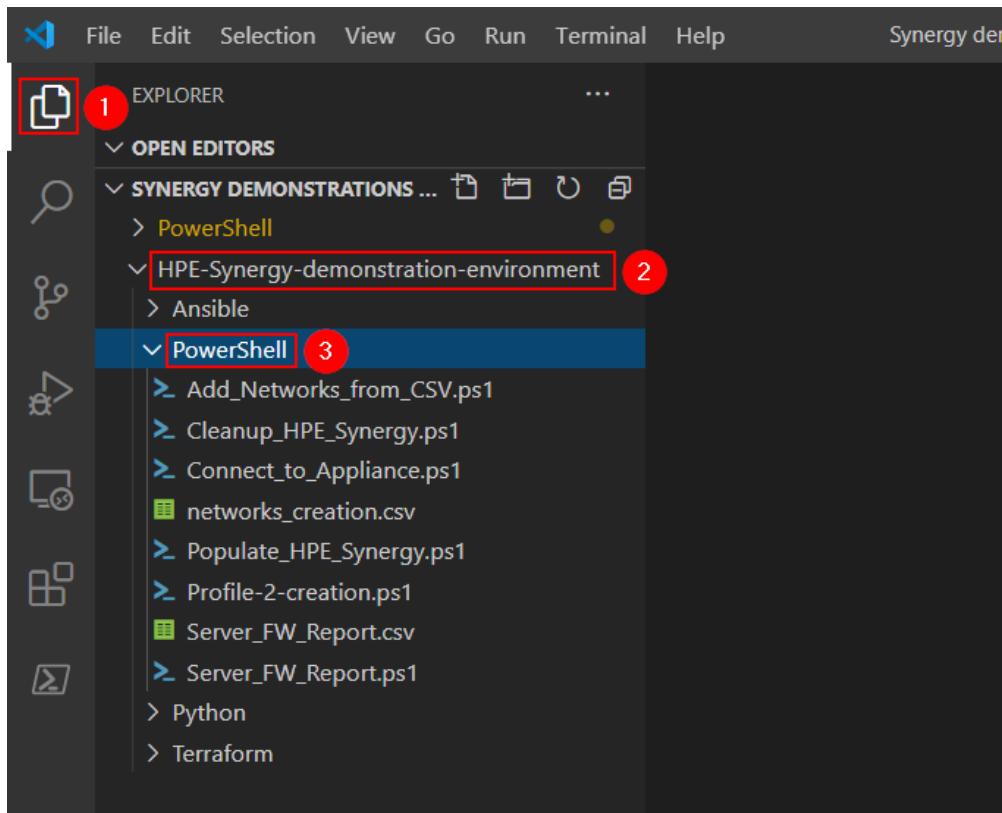


## Final appliance configuration

To configure and populate entirely the HPE OneView DCS demonstration appliance, we are going to use a PowerShell script.

**Note:** Running this script can be a good demonstration use case to show how we can fully configure a OneView/Synergy environment.

- Open the **Synergy demonstrations on Windows** workspace in VS Code
- From the explorer, select the **HPE-Synergy-demonstration-environment** repository and open the **PowerShell** folder.

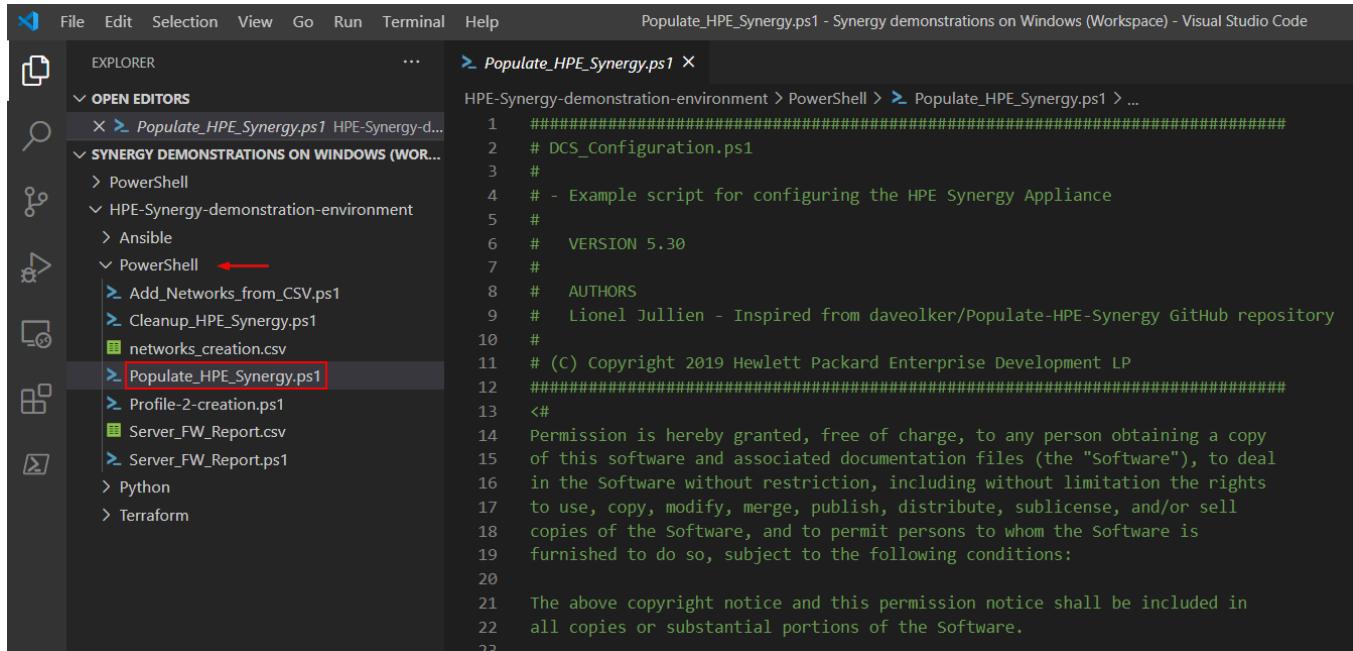


- In this folder, you can find two scripts, `Populate_HPE_Synergy.ps1` to configure and populate entirely the HPE OneView DCS demonstration appliance and `Cleanup_HPE_Synergy.ps1` to clean up everything

**Note:** These scripts were inspired by Dave Olker's work from HPE who maintains a GitHub repository, see <https://github.com/daveolker/Populate-HPE-Synergy>.

`Populate_HPE_Synergy.ps1` has been modified to meet the needs of the different scenarios proposed in this guide.

- Open **Populate\_HPE\_Synergy.ps1**



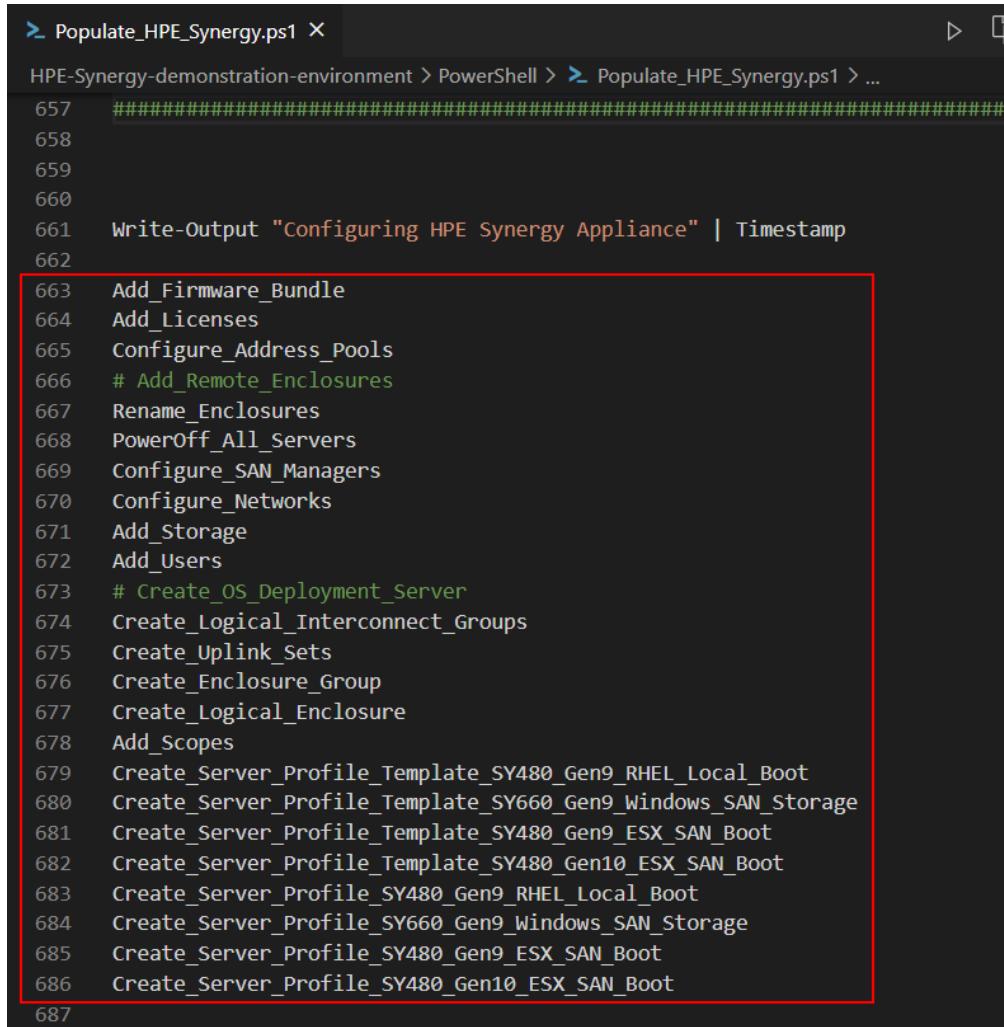
```
1 #####  
2 # DCS_Configuration.ps1  
3 #  
4 # - Example script for configuring the HPE Synergy Appliance  
5 #  
6 # VERSION 5.30  
7 #  
8 # AUTHORS  
9 # Lionel Jullien - Inspired from daveolker/Populate-HPE-Synergy GitHub repository  
10 #  
11 # (C) Copyright 2019 Hewlett Packard Enterprise Development LP  
12 #####  
13 <#  
14 Permission is hereby granted, free of charge, to any person obtaining a copy  
15 of this software and associated documentation files (the "Software"), to deal  
16 in the Software without restriction, including without limitation the rights  
17 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
18 copies of the Software, and to permit persons to whom the Software is  
19 furnished to do so, subject to the following conditions:  
20  
21 The above copyright notice and this permission notice shall be included in  
22 all copies or substantial portions of the Software.  
23
```

- This script executes the following steps:

- Prompts the user for the location of a Service Pack for ProLiant to upload as a Firmware Bundle
- Prompts the user for a text file containing OneView and Synergy 8GB Fibre Channel Licenses
- Configures two additional Synergy Enclosures
- Renames all five Synergy Enclosures
- Powers off all Compute Modules
- Configures the simulated Cisco SAN Managers
- Configures multiple Ethernet, Fibre Channel, and FCoE Networks
- Configures multiple 3PAR Storage Arrays, Volume Templates, and Volumes
- Adds various Users with different permissions
- Deploys an HPE Image Streamer OS Deployment instance
- Creates Logical Interconnect Groups
- Creates multiple Uplink Sets
- Creates an Enclosure Group
- Creates a Logical Enclosure
- Creates multiple sample Server Profile Templates
- Creates multiple sample Server Profiles
- Adds various Scopes
- Configures remote resources including: LE, LI, LIGs, Enclosure Group

**Note:** To get a better display of the script, press **CTRL + K + CTRL + O** to collapse all regions

- Scroll-down to line 663 in the script. This is where all functions defined are called:



```
Populate_HPE_Synergy.ps1 X
HPE-Synergy-demonstration-environment > PowerShell > Populate_HPE_Synergy.ps1 > ...
657 #####
658
659
660
661 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
662
663 Add_Firmware_Bundle
664 Add_Licenses
665 Configure_Address_Pools
666 # Add_Remote_Enclosures
667 Rename_Enclosures
668 PowerOff_All_Servers
669 Configure_SAN_Managers
670 Configure_Networks
671 Add_Storage
672 Add_Users
673 # Create_OS_Deployment_Server
674 Create_Logical_Interconnect_Groups
675 Create_Uplink_Set
676 Create_Enclosure_Group
677 Create_Logical_Enclosure
678 Add_Scopes
679 Create_Server_Profile_Template_SY480_Gen9_RHEL_Local_Boot
680 Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
681 Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
682 Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
683 Create_Server_Profile_SY480_Gen9_RHEL_Local_Boot
684 Create_Server_Profile_SY660_Gen9_Windows_SAN_Storage
685 Create_Server_Profile_SY480_Gen9_ESX_SAN_Boot
686 Create_Server_Profile_SY480_Gen10_ESX_SAN_Boot
687
```

Each function runs a specific task:

- `Add_Firmware_Bundle` adds a Service Pack for ProLiant ISO file to the OneView repository. This is optional but needed if you want to demonstrate firmware upgrade capabilities of HPE OneView but keep in mind that the firmware upgrade demonstration will be limited as you cannot update firmware of simulated hardware/server profiles. When run, the function asks to specify the location of a Service Pack for ProLiant ISO file.

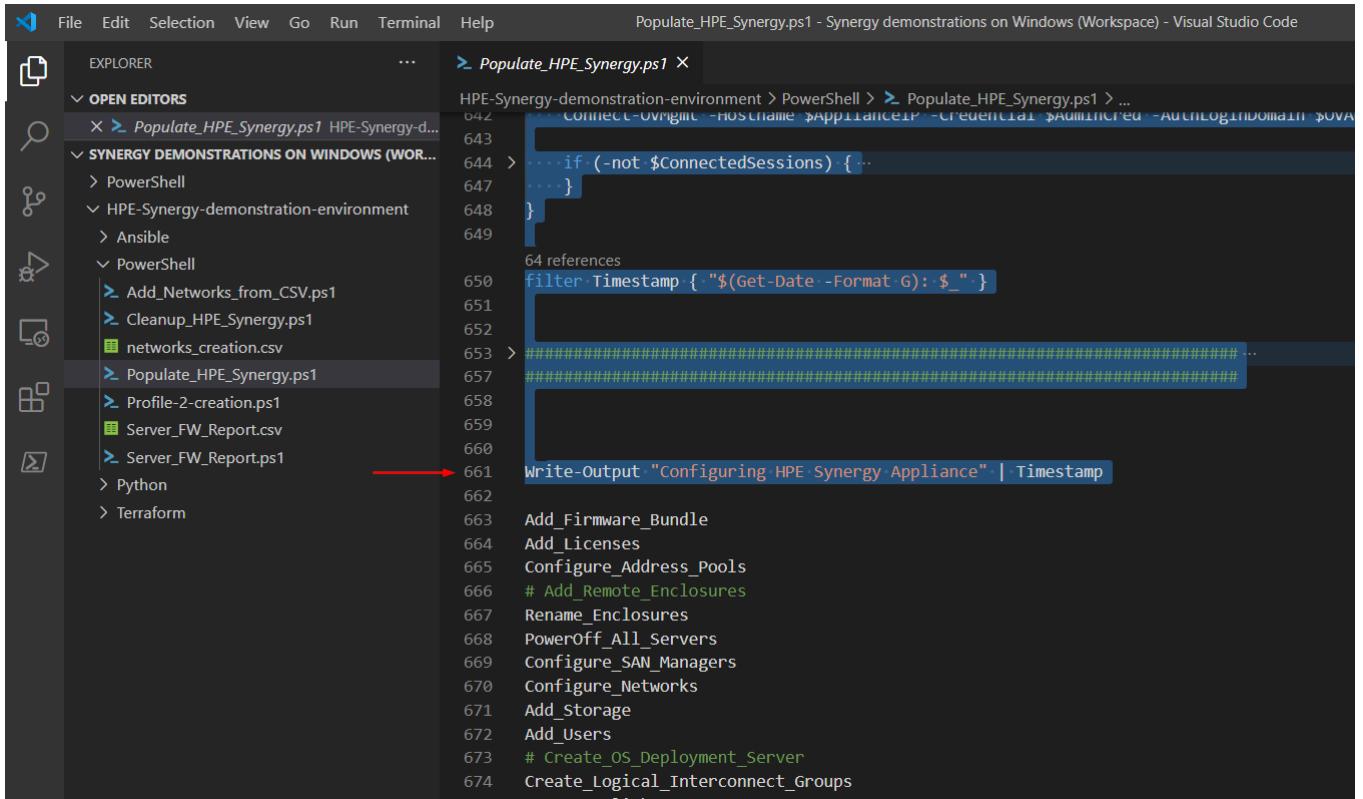
**Note:** You can download an HPE Synergy SPP from <https://www.hpe.com/downloads/synergy>

**Note:** You can always upload it later to your OneView appliance by going to Firmware Bundles in the Appliance section of the main menu.

- `Add_Licenses` adds a Synergy Fibre Channel license to OneView. When run, the function asks to specify the filename containing the license. OneView licenses is not required for a Synergy environment.
- `Configure_Address_Pools` configures the virtual address pools for MAC, WWN, and Serial Numbers.
- `Add_Remote_Enclosure` adds two additional enclosures in OneView (optional).
- `Rename_Enclosures` renames the 5 enclosures with more convenient names (e.g. Synergy-Encl-1, Synergy-Encl-2, etc.)
- `PowerOff_All_Servers` turns all servers off to prepare the creation of Server Profiles
- `Configure_SAN_Managers` adds two Cisco MDS 9250i switches
- `Configure_Networks` creates 15 networks (Ethernet, FC and FCoE)
- `Add_Storage` adds 2 x 3PAR 7200 Storage Systems and 3 x StoreVirtual then adds 3 x Volumes and 7 x Volume Templates
- `Add_Users` creates 5 new users with different roles.
- `Create_OS_Deployment_Server` configures the enclosures for Image Streamer
- `Create_Logical_Interconnect_Groups` creates 3 x LIGs (SAS, FC and FlexFabric)
- `Create_Uplink_Sets` configures 8 x uplink sets (2xFC, 2xFCoE, 1xMgmt, 1x vMotion, 1xImageStreamer, 1xProd)
- `Create_Enclosure_Group` creates an EG with 3 frames/LIGs + Image Streamer
- `Create_Logical_Enclosure` creates the LE with EG/LIGs/Streamer configured previously
- `Add_Scopes` creates a new scope with the first frame and the production networks
- `Create_Server_Profile_xxx` creates different server profiles types using Gen9 and Gen10 servers



- Now select the first part of the script from line 1 to 661



The screenshot shows the Visual Studio Code interface with the file 'Populate\_HPE\_Synergy.ps1' open in the editor. The code is a PowerShell script for HPE Synergy demonstration. A red arrow points to line 661, which contains the command 'Write-Output "Configuring HPE Synergy Appliance" | Timestamp'. The code includes various functions like 'Add\_Networks\_from\_CSV.ps1', 'Cleanup\_HPE\_Synergy.ps1', and 'networks\_creation.csv'. The script also handles session connection and timestamping.

```
642     Connect-UVRegion -Hostname $APPLIANCEIP -Credential $AdminUserCred -AuthLogInDomain $UVADomain
643
644     > if (-not $ConnectedSessions) { ...
645     ...
646 }
647
648 }
649
650 filter Timestamp { "$(Get-Date -Format G): $_" }
651
652
653 > #####
654 #####
655 #####
656 #####
657 #####
658 #####
659 #####
660 #####
661 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
662
663 Add_Firmware_Bundle
664 Add_Licenses
665 Configure_Address_Pools
666 # Add_Remote_Enclosures
667 Rename_Enclosures
668 PowerOff_All_Servers
669 Configure_SAN_Managers
670 Configure_Networks
671 Add_Storage
672 Add_Users
673 # Create_OS_Deployment_Server
674 Create_Logical_Interconnect_Groups
```

- Once selected, press **F8** (or right-click **Run Selection**) to execute only the selected lines:

The screenshot shows the Visual Studio Code interface with a PowerShell script file open. A context menu is displayed over a selected block of code. The 'Run Selection' option is highlighted with a red box. Other options visible in the menu include 'Go to Definition', 'Go to References', 'Peek', 'Find All References', 'Change All Occurrences', 'Format Document', 'Format Selection', 'Get Help for Command', 'Cut', 'Copy', 'Paste', and 'Command Palette...'. The script file contains various cmdlets like Write-Output, Add\_Firmware\_Bundle, and Configure\_Networks.

- As requested, enter the Composer IP: **192.168.56.101**
- Then **Administrator / password** for the OneView credentials

The screenshot shows the PowerShell Integrated Terminal in Visual Studio Code. The terminal window title is '2: PowerShell Integrated'. The output shows the execution of a PowerShell script. It prompts for the OneView administrator username ('Administrator') and password ('\*\*\*\*\*'). It then lists the connection details, including the connection ID (1), name (192.168.56.101), user name (Administrator), authentication domain (Local), and default status (True). The timestamp indicates the command was run on 1/17/2020 at 3:14:26 PM. The terminal also displays a watermark for 'Activate Windows' and 'Go to Settings to activate Windows.'

- Make sure no error is thrown.

- Then select and run each function one at a time from line **663** to **680** by pressing **F8** and move back and forth between VS Code and OneView web interface to validate the configuration change of each step.

```
> Populate_HPE_Synergy.ps1 <
HPE-Synergy-demonstration-environment > PowerShell > > Populate_HPE_Synergy.ps1 > ...
660
661     Write-Output "Configuring HPE Synergy Appliance" | Timestamp
662
663     Add_Firmware_Bundle 1 2 F8
664     Add_Licenses
665     Configure_Address_Pools
666     # Add_Remote_Enclosures
667     Rename_Enclosures
668     PowerOff_All_Servers
669     Configure_SAN_Managers
670     Configure_Networks
671     Add_Storage
672     Add_Users
673     # Create_OS_Deployment_Server
674     Create_Logical_Interconnect_Groups
675     Create_Uplink_Sets
676     Create_Enclosure_Group
677     Create_Logical_Enclosure
678     Add_Scopes
679     Create_Server_Profile_Template_SY480_Gen9_RHEL_Local_Boot
680     Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
681     Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
682     Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
683     Create_Server_Profile_SY480_Gen9_RHEL_Local_Boot
684     Create_Server_Profile_SY660_Gen9_Windows_SAN_Storage
685     Create_Server_Profile_SY480_Gen9_ESX_SAN_Boot
686     Create_Server_Profile_SY480_Gen10_ESX_SAN_Boot
687
```

**Note:** Only the first two steps (`Add_Firmware_Bundle` and `Add_License`) require some input. They are optional.

**Note:** `Add_Remote_Enclosures` takes long minutes to complete and is also optional, hence the comment. We do not use any of the remote enclosures in the demonstration scenarios available in this guide.

- At the creation of the Logical Enclosure, the script should run successfully, but the UI will show an error message if no FC license has been added during the Add\_Licenses step:

The screenshot shows the HPE OneView interface. On the left, under 'Logical Enclosures 1', there is a green button labeled '+ Create logical enclosure'. A list shows one item: 'LE-Synergy-Local' with a green dot next to it. On the right, the details for 'LE-Synergy-Local' are shown. At the top, there is a red banner with the following text:  
**Create Error 8m18s**  
 Validate firmware.  
 Configure servers and create logical interconnects.  
 Update enclosure information.  
**Issue** Encountered problems creating logical interconnects: LE-Synergy-Local-LIG-FlexFabric.  
**Resolution** View the subtask details for the listed logical interconnects to see the problems and recommended actions.

This is due to the lack of license for the Fibre Channel connectivity. Licenses are required with the Virtual Connect SE 40Gb F8 and 100G F32 Modules. Without these licenses, FC ports cannot be activated but this is not blocking any operations for our environment.

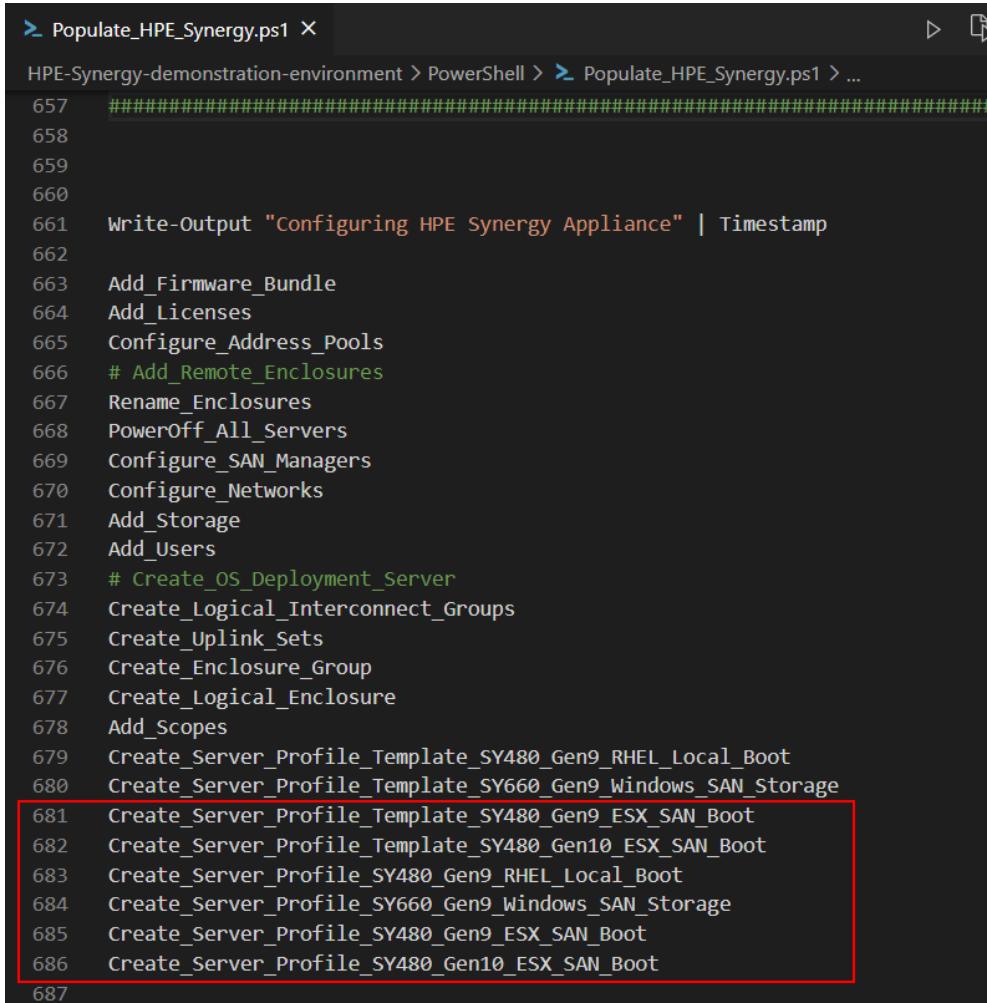
The following error is displayed during the creation of the Logical Enclosure:

The screenshot shows a task log entry for 'Assign FC upgrade licenses'. The task was started at 1/16/20 2:33:35 pm and completed at 1/16/20 2:33:53 pm with an error status (2m16s). The task details are:  
**Assign FC upgrade licenses**      LE-Synergy-Local-LIG-FlexFabric      1/16/20 2:33:35 Error 2m16s  
 pm  
 Assign 'Synergy 8Gb FC Upgrade' license for  
Synergy-Encl-1, interconnect 3  
Synergy-Encl-2, interconnect 6  
**Issue** There were no 'Synergy 8Gb FC Upgrade' licenses available on the appliance.  
**Resolution** Add 'Synergy 8Gb FC Upgrade' license keys to the appliance. If sufficient licenses are added for all interconnects requiring them, licenses will be automatically applied to all of them. If not, reapply the logical interconnect configuration.  
[Add license](#)  
[Reapply configuration](#)  
[Licenses](#)

A warning message is also displayed on the Logical Interconnect resource:

The screenshot shows the 'Logical Interconnects 7' page. A yellow banner at the top states: ▲ There were no 'Synergy 8Gb FC Upgrade' licenses available on the appliance. Active. The timestamp is 1/16/20 2:35:53 pm. Below the banner, the 'Logical Interconnect' section displays six items:  
 1. Internal: no networks  
 2. US-SAN-...: 1 network, 1 uplink port  
 3. US-ESX-...: 1 network, 2 uplink ports  
 4. US-ESX-v...: 1 network, 2 uplink ports  
 5. US-Prod: 4 networks, 2 uplink ports  
 6. US-SAN-...: 1 network, 1 uplink port

- After the creation of the second server profile Template at line 680, you can stop. The other profile creations and setup of remote enclosures will be kept for customer demonstration.



```
> Populate_HPE_Synergy.ps1 <
HPE-Synergy-demonstration-environment > PowerShell > > Populate_HPE_Synergy.ps1 > ...
657 #####
658
659
660
661 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
662
663 Add_Firmware_Bundle
664 Add_Licenses
665 Configure_Address_Pools
666 # Add_Remote_Enclosures
667 Rename_Enclosures
668 PowerOff_All_Servers
669 Configure_SAN_Managers
670 Configure_Networks
671 Add_Storage
672 Add_Users
673 # Create_OS_Deployment_Server
674 Create_Logical_Interconnect_Groups
675 Create_Uplink_Sets
676 Create_Enclosure_Group
677 Create_Logical_Enclosure
678 Add_Scopes
679 Create_Server_Profile_Template_SY480_Gen9_RHEL_Local_Boot
680 Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
681 Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
682 Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
683 Create_Server_Profile_SY480_Gen9_RHEL_Local_Boot
684 Create_Server_Profile_SY660_Gen9_Windows_SAN_Storage
685 Create_Server_Profile_SY480_Gen9_ESX_SAN_Boot
686 Create_Server_Profile_SY480_Gen10_ESX_SAN_Boot
687
```

- For the last step, enter the following command in the console to disconnect VS Code from the appliance:

```
Disconnect-OVMgmt
```

## Creation of a final setup snapshot

Once the setup is complete, we can create a snapshot to save the appliance final configuration.

Before creating the second snapshot:

- Try to resolve any issues that may be reported by the appliance

**Note:** The Logical Enclosure inconsistent error is expected as we do not have any Synergy 8Gb FC Upgrade licenses.

The screenshot shows the HPE OneView interface. On the left, there's a sidebar with a 'Logical Enclosures' section containing a button to 'Create logical enclosure'. The main panel is titled 'LE-Synergy-Local' and shows an 'Overview' tab. A yellow warning bar at the top says: 'The logical enclosure is inconsistent with its enclosure group EG-Synergy-Local. Active'. Below this, there's a 'General' section with details about consistency state, enclosure group (EG-Synergy-Local), enclosures (Synergy-Encl-1, Synergy-Encl-2, Synergy-Encl-3), and logical interconnects (various LIG entries). The date and time '2/19/20 2:06:41 pm' are also visible.

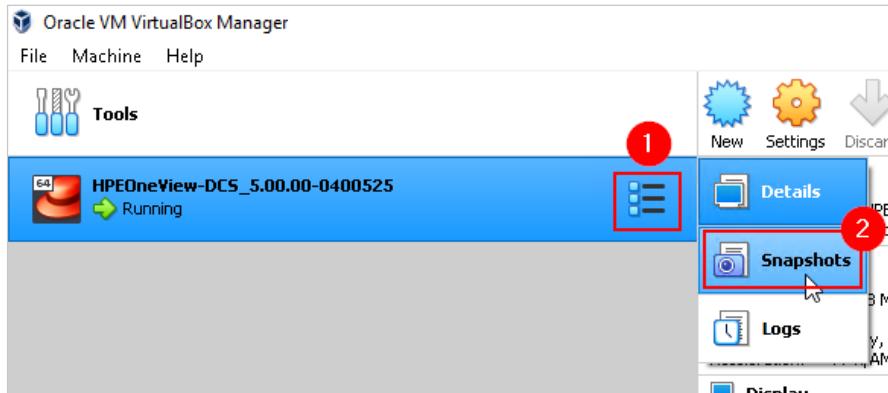
- Make sure there is no OneView tasks that is still running. Go in OneView / **Activity** then use **Running** State in the Filter tool:

The screenshot shows the 'Activity' page in HPE OneView. The search bar at the top has 'state:running' entered. A red box highlights the 'Running' option in a dropdown menu on the right. The main table shows columns for Name, Resource, Date, State, and Owner. A message 'No matches' is displayed. The 'State' column header has a dropdown arrow, indicating it can be sorted or filtered.

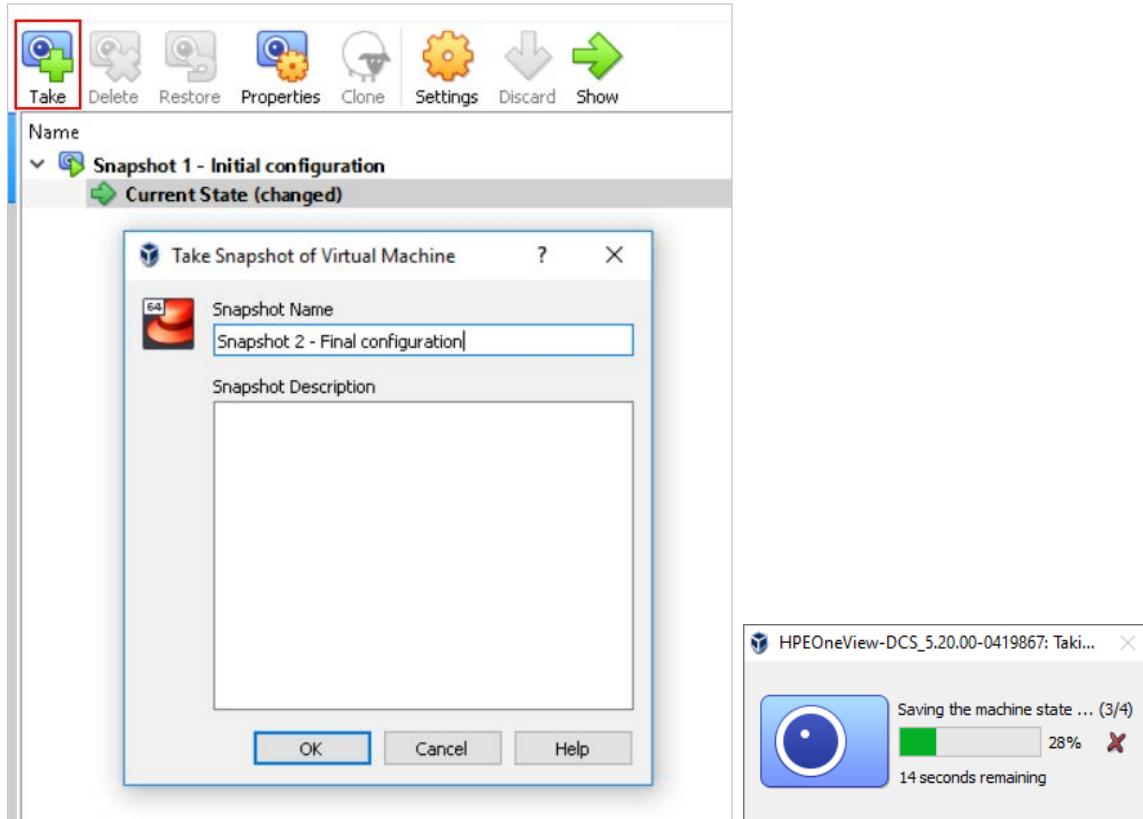
**Note:** Taking a snapshot while the appliance is running is a key practice to avoid waiting long minutes for the appliance to start.

To create a snapshot to save the appliance final configuration:

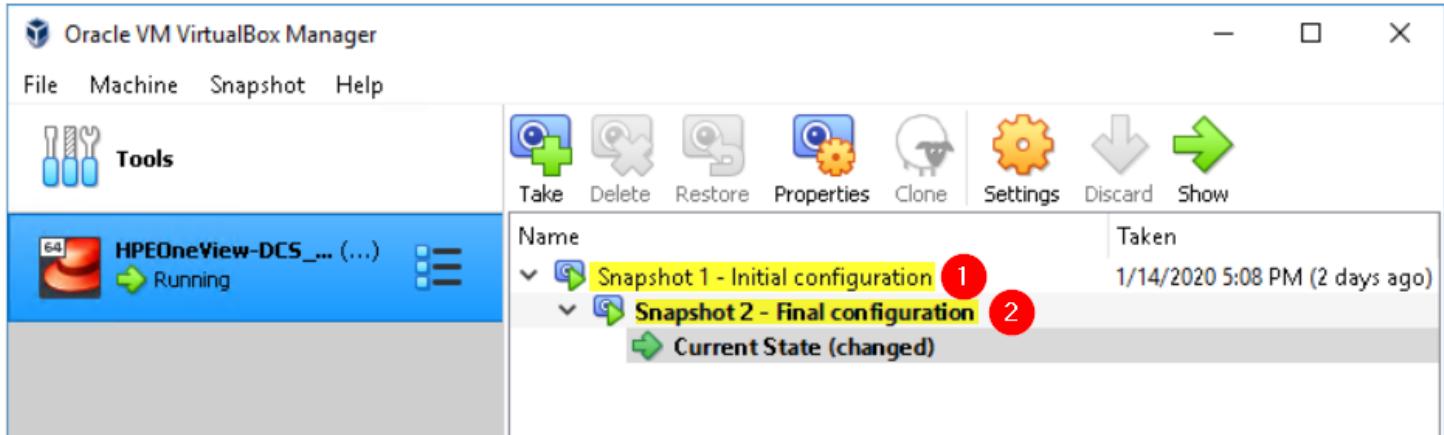
- Go to the VirtualBox interface, select the **Tools** menu then **Snapshots**



- Then press on **Take**, enter a snapshot name like **Snapshot 2 – Final configuration** then click **OK**



To sum up, we have now two snapshots:



- 1- First snapshot to show how to automate the setup of Synergy and the power of our infrastructure as code implementation. OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network).
- 2- Second snapshot is with a fully configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

## Chapter-7 – Preparing the live demonstration scenarios

### Demonstrating Synergy Composer and OneView Key features

If you simply want to highlight the Synergy/OneView key features, you can refer to the latest *HPE OneView Deployment and Management Guide* you can find in the [OneView documentation](#).

### Demonstrating Infrastructure programmability

To demonstrate Software-Defined infrastructure, infrastructure programmability and total datacenter automation with OneView/Synergy and positively impact our customers, we need some good preparation.

If you need to introduce the REST API, show the resource model, show a typical OneView object content, etc. we usually recommend the use of Postman. In *Postman - Scenario 1* we provide some guidelines to drive you into the REST API introduction speech.

To demonstrate any aspects of the Software Defined Infrastructure, you can use the tons of PowerShell scripts available on various GitHub repos, however, to help you, we have built three PowerShell scenarios showing:

- A day-to-day operation task automation
- A report creation
- Accelerating a configuration change.

For Python, we have two scenarios:

- A script to automate the creation of a server profile
- A report creation
- Accelerating a configuration change.

For Ansible, we have three scenarios:

- A playbook to collect information in OneView
- A playbook to automate the provisioning of several servers
- A playbook to unprovision several servers automatically

For Terraform, we have three scenarios:

- A configuration file to accelerate a configuration change (add a new network)
- A configuration file to automate the provisioning of a server profile
- A configuration file to automate the unprovisioning of a server profile

More scenarios will be added over time...



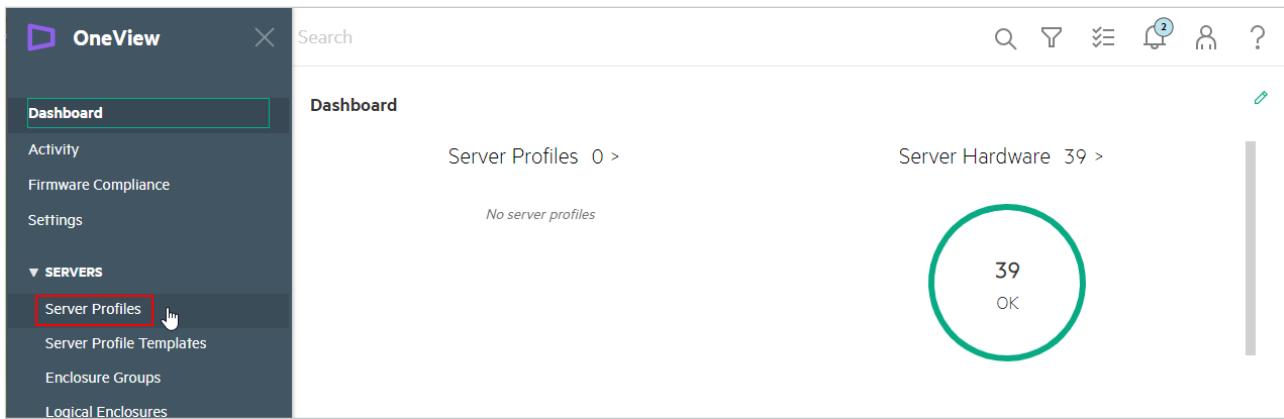
## Postman - Scenario 1 – Introduction to the OneView REST API

### PowerShell – Scenario 1 - Day-to-day operation task automation

In this scenario, we are going to show how easy it is to generate script code from existing OneView resources. The code generated is a starting point to be used for repeating similar tasks performed by the UI, or to incorporate into scripts or workflows.

**Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a configured Logical Enclosure is available)

- Open OneView GUI, log in using **Administrator / password**
- Go to **Server Profiles**



- Click on **Create profile**

- Name the profile **Profile-1**, select the Server Profile Template **HPE Synergy 480 Gen9 with Local Boot for RHEL Template** and use the server hardware **Synergy-Encl-1, bay 5**:

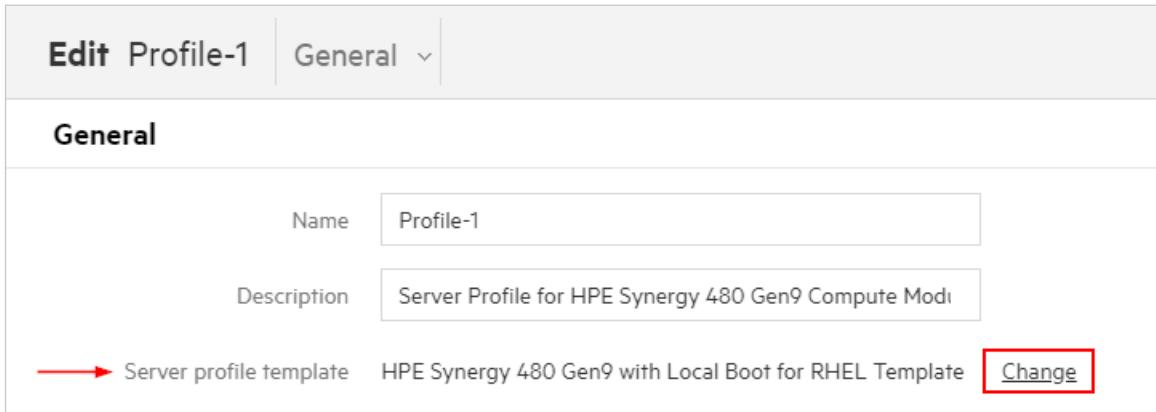
The screenshot shows the 'Create Server Profile' dialog box. The 'General' tab is selected. A yellow warning box at the top states: "On next assignment to server hardware, the local storage controllers marked for re-initialization will have their logical drives deleted making existing data inaccessible. It is strongly suggested that you back up any data on existing logical drives on these controllers before applying a profile with this option selected. If you wish to preserve any existing logical drives on these controllers, deselect the re-initialize storage checkbox." The 'Name' field contains 'Profile-1'. The 'Server profile template' field is set to 'HPE Synergy 480 Gen9 with Local Boot for RHEL Temp' (with 'Temp' in red). The 'Server hardware' field is set to 'Synergy-Encl-1, bay 5'. Other fields include 'Scope' (empty), 'Description' (partial text), 'Server hardware type' (SY 480 Gen9 1), 'Enclosure group' (EG-Synergy-Local), and 'Affinity' (Device bay).

- Then click **Create**

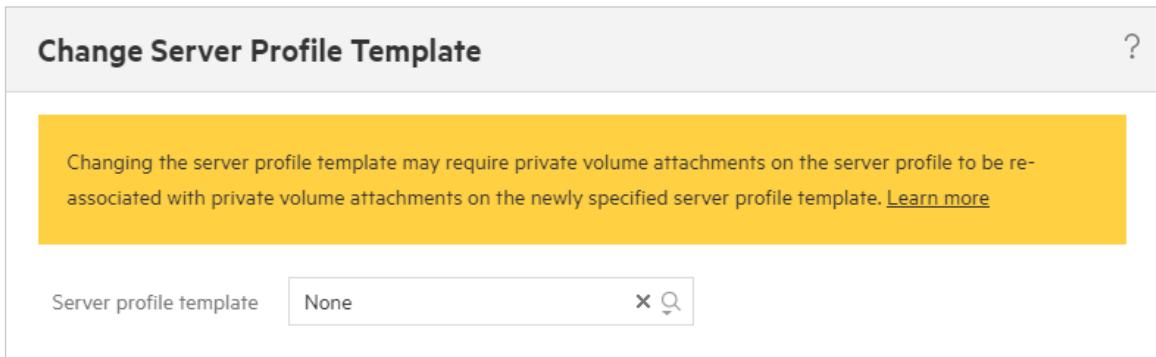
Once created, it is necessary to unlink the server Profile from the Server Profile Template to get the most from the `ConvertTo-OVPowerShellScript` cmdlet that we are going to use next.



- **Edit** the Profile then click **Change** remove the Server Profile Template



- Then select **None** in the dropdown menu and click **OK** twice



- Open VS Code using the **Synergy demonstrations on Windows** workspace
- Open `Connect_to_Appliance.ps1` located in the **PowerShell** folder of the **HPE-Synergy-demonstration-environment** repository:

The screenshot shows the Visual Studio Code interface with the 'Connect\_to\_Appliance.ps1' script open in the editor. The file content is as follows:

```
#IP address of OneView
$IP = "192.168.56.101"

# OneView Credentials
$username = "Administrator"
$password = "password"

$secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)

Connect-OVMgmt -appliance $IP -Credential $credentials
```

- This script allows the VS Code console to connect to the appliance. Press **F5** to run the script.
- A OneView connection confirmation should be displayed in the Console, now we can execute any cmdlets from the OneView library:

```
PS C:\VS Code projects\PowerShell> c:\VS Code projects\Repositories\HPE-Synergy-demonstration-environment\PowerShell\Connect_to_Appliance.ps1
This management appliance is a company owned asset and provided for the exclusive use of authorized personnel. Unauthorized use or abuse of this system may lead to corrective action including termination, civil and/or criminal penalties.

ConnectionID Name          UserName      AuthLoginDomain Default
----- ----          -----      -----          -----
1         192.168.56.101  Administrator  LOCAL          True
```

- We can use the `get-ovserverprofile` cmdlet to get the list of server profiles. To reduce the response to our new server profile, we can enter:

```
get-ovserverprofile -name Profile-1
```

```
PS C:\VS Code projects\Repositories\HPE-Synergy-demonstration-environment\PowerShell> get-ovserverprofile -name Profile-1

Name      Status Compliance Template                                Server Hardware      Server Hardware Type Enclosure Group Affinity
----      ----      -----      -----                                -----      -----      -----      -----      -----      -----
Profile-1 OK      Compliant HPE Synergy 480 Gen9 with Local Boot for RHEL Template Synergy-Encl-1, bay 5 SY 480 Gen9 1      EG-Synergy-Local Bay

PS C:\VS Code projects\Repositories\HPE-Synergy-demonstration-environment\PowerShell>
```

- The next step is to use the `ConvertTo-OVPowerShellScript` to generate the required lines of code to produce this server profile.

```
Get-OVServerProfile -Name Profile-1 | ConvertTo-OVPowerShellScript
```

```
PS C:\VS Code projects\Repositories\HPE-Synergy-demonstration-environment\PowerShell> get-ovserverprofile -name Profile-1 | ConvertTo-OVPowerShellScript

# ----- Attributes for ServerProfile "Profile-1"
$name          = "Profile-1"
$description   = "Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL"
$server        = Get-OVServer -Name "Synergy-Encl-1, bay 5"
$affinity      = "Bay"
# ----- Connections section
# ----- Attributes for connection "1"
$connID        = 1
$connName      = "Mgmt-1"
$connType      = "Ethernet"
$netName       = "Mgmt"
$ThisNetwork   = Get-OVNetwork -Type Ethernet -Name $netName
$portID        = "Mezz 3:1-a"
$requestedMbps = 2500
$conn1         = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connType -Network $ThisNetwork -PortId $portID -RequestedBw $requestedMbps
# ----- Attributes for connection "2"
$connID        = 2
$connName      = "Mgmt-2"
$connType      = "Ethernet"
$netName       = "Mgmt"
$ThisNetwork   = Get-OVNetwork -Type Ethernet -Name $netName
$portID        = "Mezz 3:2-a"
$requestedMbps = 2500
$conn2         = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connType -Network $ThisNetwork -PortId $portID -RequestedBw $requestedMbps
# ----- Attributes for connection "3"
$connID        = 3
$connName      = "Prod-NetworkSet-1"
$connType      = "Ethernet"
$netName       = "Prod"
```

This cmdlet can assist administrators or scripters to help generate script code from specific resources. The code generated is a starting point to be used for repeating similar tasks performed by the UI, or to incorporate into scripts or workflows.

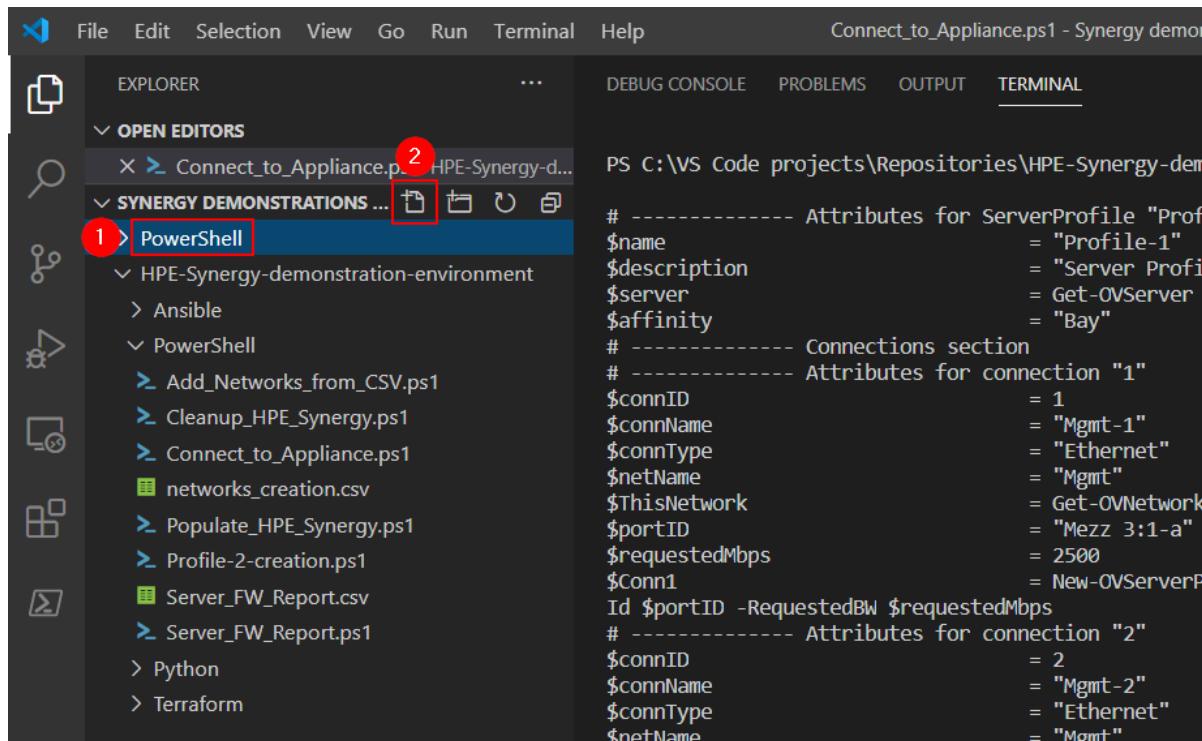
**Note:** Not all resources are supported by `ConvertTo-OVPowerShellScript`, you can use the `Get-Help` command to see the list of supported resources: `Get-help ConvertTo-OVPowerShellScript`

- Lastly, disconnect the PowerShell client to the appliance:

```
Disconnect-OVMgmt
```

- Next, we can demonstrate how we can easily create a script using this generated code to create a new Server Profile in OneView. This will require just a few modifications.

Select your PowerShell folder (not the clone one from the GitHub repo) then click on **New File** to create a new script:



```
PS C:\VS Code projects\Repositories\HPE-Synergy-demonstration-environment> # ----- Attributes for ServerProfile "Profile-1"
$Name = "Profile-1"
$Description = "Server Profile"
$Server = Get-OVServer
$Affinity = "Bay"
# ----- Connections section
# ----- Attributes for connection "1"
$connID = 1
$connName = "Mgmt-1"
$connType = "Ethernet"
$NetName = "Mgmt"
$ThisNetwork = Get-OVNetwork
$PortID = "Mezz 3:1-a"
$RequestedMbps = 2500
$Conn1 = New-OVServerPort
$Id $PortID -RequestedBW $RequestedMbps
# ----- Attributes for connection "2"
$connID = 2
$connName = "Mgmt-2"
$connType = "Ethernet"
$NetName = "Mgmt"
```

- Name it **Profile-2-creation.ps1**.

**Note:** Make sure you name the new file with a **.ps1** extension.

- Copy/paste the code generated by `ConvertTo-OVPowerShellScript` in this new PowerShell script

```

File Edit Selection View Go Run Terminal Help • Profile-2-creation.ps1 - Synergy demonstrations on Windows (Workspace) - Visual Studio Code
EXPLORER OPEN EDITORS 1 UNSAVED
  > Connect_to_Appliance.ps1 HPE-Synergy-d...
  ● > Profile-2-creation.ps1 PowerShell
SYNTERGY DEMONSTRATIONS ON WINDOWS (WORKSPACE)
  > PowerShell
    > Profile-2-creation.ps1
  > HPE-Synergy-demonstration-environment
    > Ansible
    > PowerShell
      > Add_Networks_from_CSV.ps1
      > Cleanup_HPE_Synergy.ps1
      > Connect_to_Appliance.ps1
      > networks_creation.csv
      > Populate_HPE_Synergy.ps1
      > Profile-2-creation.ps1
      > Server_FW_Report.csv
      > Server_FW_Report.ps1
    > Python
    > Terraform
PowerShell > > Profile-2-creation.ps1 > ...
1 # ----- Attributes for ServerProfile "Profile-1"
2 $name = "Profile-1"
3 $description = "Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Storage"
4 $server = Get-OVServer -Name "Synergy-Encl-1, bay 5"
5 $affinity = "Bay"
6 # ----- Connections section
7 # ----- Attributes for connection "1"
8 $connID = 1
9 $connName = "Mgmt-1"
10 $connType = "Ethernet"
11 $netName = "Mgmt"
12 $thisNetwork = Get-OVNetwork -Type Ethernet -Name $netName
13 $portID = "Mezz 3:1-a"
14 $requestedMbps = 2500
15 $conn1 = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -Network $thisNetwork -PortID $portID -RequestedMbps $requestedMbps
16 # ----- Attributes for connection "2"
17 $connID = 2
18 $connName = "Mgmt-2"
19 $connType = "Ethernet"
20 $netName = "Mgmt"
21 $thisNetwork = Get-OVNetwork -Type Ethernet -Name $netName
22 $portID = "Mezz 3:2-a"
23 $requestedMbps = 2500
24 $conn2 = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -Network $thisNetwork -PortID $portID -RequestedMbps $requestedMbps
25 # ----- Attributes for connection "3"
26 $connID = 3
27 $connName = "Prod-NetworkSet-1"
28 $connType = "Ethernet"

```

- Then we need to add the commands to connect to the appliance. Add the following at the beginning of the script:

```

#IP address of OneView
$IP = "192.168.56.101"

# OneView Credentials
$username = "Administrator"
$password = "password"

$secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)

Connect-OVVmgt -appliance $IP -Credential $credentials

```

- Then change the following values:

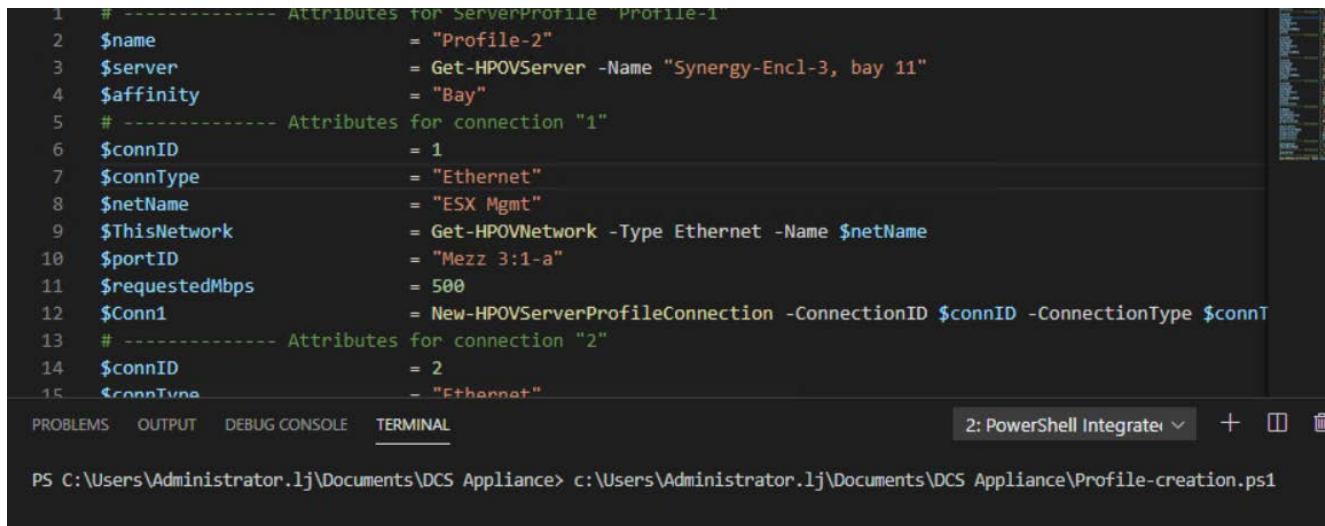
- \$name with **Profile-2**
- \$server with **Synergy-Encl-2, bay 5**
- The requested bandwidth of the management connection **1** and **2** to **1000 Mbps**

```
➤ Connect_to_Appliance.ps1    ➤ Profile-2-creation.ps1 ●
PowerShell > ➤ Profile-2-creation.ps1 > ...
8     $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
9     $credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
10
11    Connect-OVMgmt -appliance $IP -Credential $credentials
12
13    # ----- Attributes for ServerProfile "Profile[2]"
14    $name                = "Profile[2]"
15    $description         = "Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for"
16    $server              = Get-OVServer -Name "Synergy-Encl[2], bay 5"
17    $affinity             = "Bay"
18
19    # ----- Connections section
20    # ----- Attributes for connection "1"
21    $connID              = 1
22    $connName             = "Mgmt-1"
23    $connType              = "Ethernet"
24    $netName              = "Mgmt"
25    $ThisNetwork           = Get-OVNetwork -Type Ethernet -Name $netName
26    $portID               = "Mezz 3:1-a"
27    $requestedMbps         = 1000
28    $Conn1                = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -Connect
29
30    # ----- Attributes for connection "2"
31    $connID              = 2
32    $connName             = "Mgmt-2"
33    $connType              = "Ethernet"
34    $netName              = "Mgmt"
35    $ThisNetwork           = Get-OVNetwork -Type Ethernet -Name $netName
36    $portID               = "Mezz 3:2-a"
37    $requestedMbps         = 1000
38    $Conn2                = New-OVServerProfileConnection -ConnectionID $connID -Name $connName -Connect
39    # ----- Attributes for connection "3"
40    $connID              = 3
41    $connName             = "Prod-NetworkSet-1"
```

- Then add the following line at the end of the script to properly disconnect the console to the appliance:

```
Disconnect-OVMgmt
```

- Once completed, save the file by pressing **CTRL + S** then press **F5** to run the script.



```

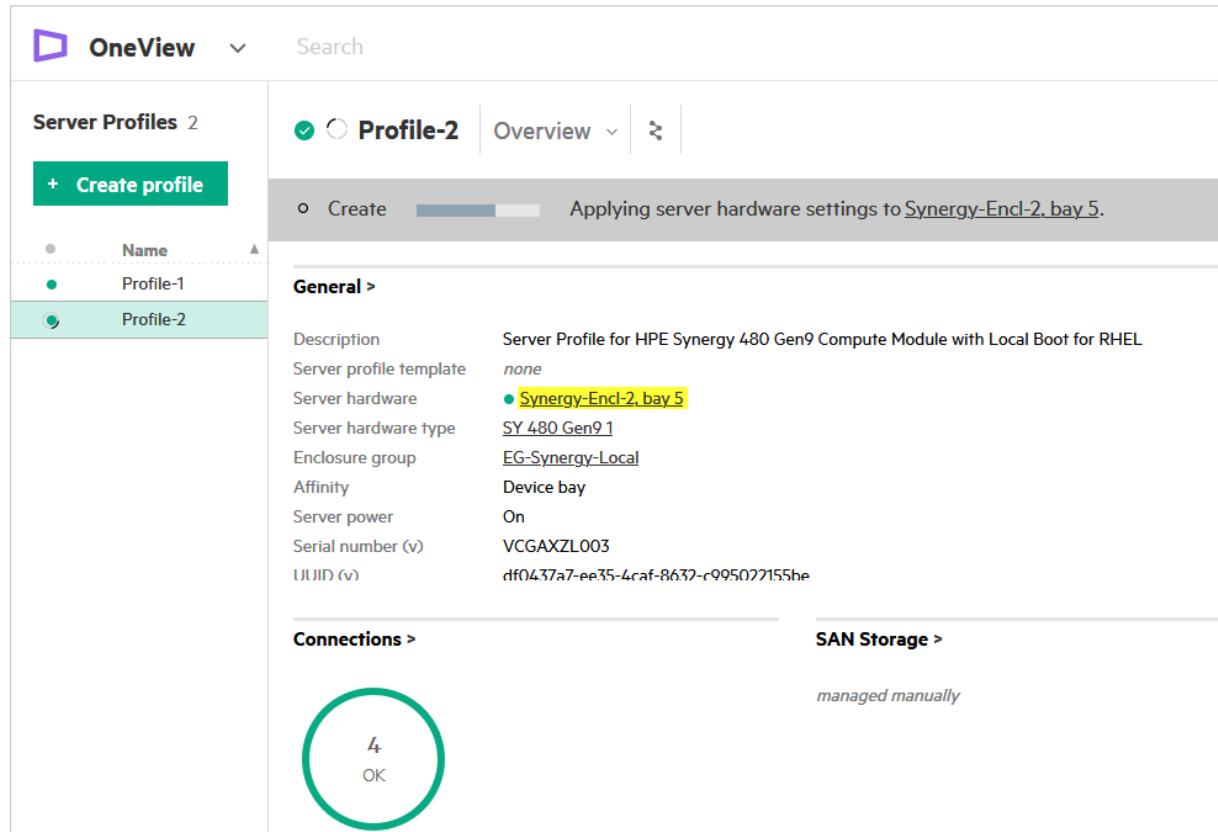
1 # ----- Attributes for ServerProfile "Profile-1"
2 $name          = "Profile-2"
3 $server        = Get-HPOVServer -Name "Synergy-Encl-3, bay 11"
4 $affinity      = "Bay"
5 # ----- Attributes for connection "1"
6 $connID        = 1
7 $connType      = "Ethernet"
8 $netName       = "ESX Mgmt"
9 $ThisNetwork   = Get-HPOVNetwork -Type Ethernet -Name $netName
10 $portID       = "Mezz 3:1-a"
11 $requestedMbps = 500
12 $Conn1         = New-HPOVServerProfileConnection -ConnectionID $connID -ConnectionType $connT
13 # ----- Attributes for connection "2"
14 $connID        = 2
15 $connType      = "Ethernet"

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: PowerShell Integrated + [ ]

PS C:\Users\Administrator.lj\Documents\DCS Appliance> c:\Users\Administrator.lj\Documents\DCS Appliance\Profile-creation.ps1

- You can then jump to the OneView GUI to show the creation of Profile-2



**Server Profiles 2**

+ Create profile

Name: Profile-2

General >

Description: Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL

Server profile template: none

Server hardware: Synergy-Encl-2, bay 5

Server hardware type: SY 480 Gen9.1

Enclosure group: EG-Synergy-Local

Affinity: Device bay

Server power: On

Serial number (v): VCGAXZL003

UUID (v): df0437a7-ee35-4caf-8632-c995022155he

Connections > SAN Storage >

4 OK managed manually

As illustrated above, the profile has been assigned to **Synergy-Encl-2, bay 5** as programmed in our code

- To show the new profile parameters that have been changed, click on the **Profile-2**, select **Connections** in the profile menu, expand connection **1** then show the 1000Mbps (1Gps) requested bandwidth that was defined in our code:

The screenshot shows the HPE OneView interface. On the left, the 'Server Profiles' list shows two profiles: 'Profile-1' and 'Profile-2'. 'Profile-2' is selected and highlighted with a red box and a red number '1'. In the center, the 'Connections' section is displayed under 'Profile-2'. A red box highlights the 'Connections' dropdown menu item, which is also labeled with a red number '2'. Below it, a progress bar indicates 'Applying server hardware settings to Synergy-Encl-2, bay 5.' The 'Connections' table lists network interfaces for Profile-2. The first interface, 'Mgmt-1', has its 'Requested bandwidth' set to '1 Gb/s' (highlighted with a red box and red number '4'). Other details for 'Mgmt-1' include: Interconnect Type: Ethernet; MAC address: C6:6C:1B:70:00:04 (v); Requested virtual functions: None; Max bandwidth: 20 Gb/s; Link aggregation group: None. Other listed connections include 'Mgmt-2', 'Prod-NetworkSet-1', and 'Prod-NetworkSet-2'.

ID	Name	Network	Port	Boot
1	Mgmt-1	Mgmt VLAN100 Synergy-Encl-1.interconnect_3	Mezzanine 3:1-a	Not bootable
2	Mgmt-2	Mgmt VLAN100	Mezzanine 3:2-a	Not bootable
3	Prod-NetworkSet-1	Prod (network set)	Mezzanine 3:1-c	Not bootable
4	Prod-NetworkSet-2	Prod (network set)	Mezzanine 3:2-c	Not bootable

This concludes PowerShell – Scenario 1

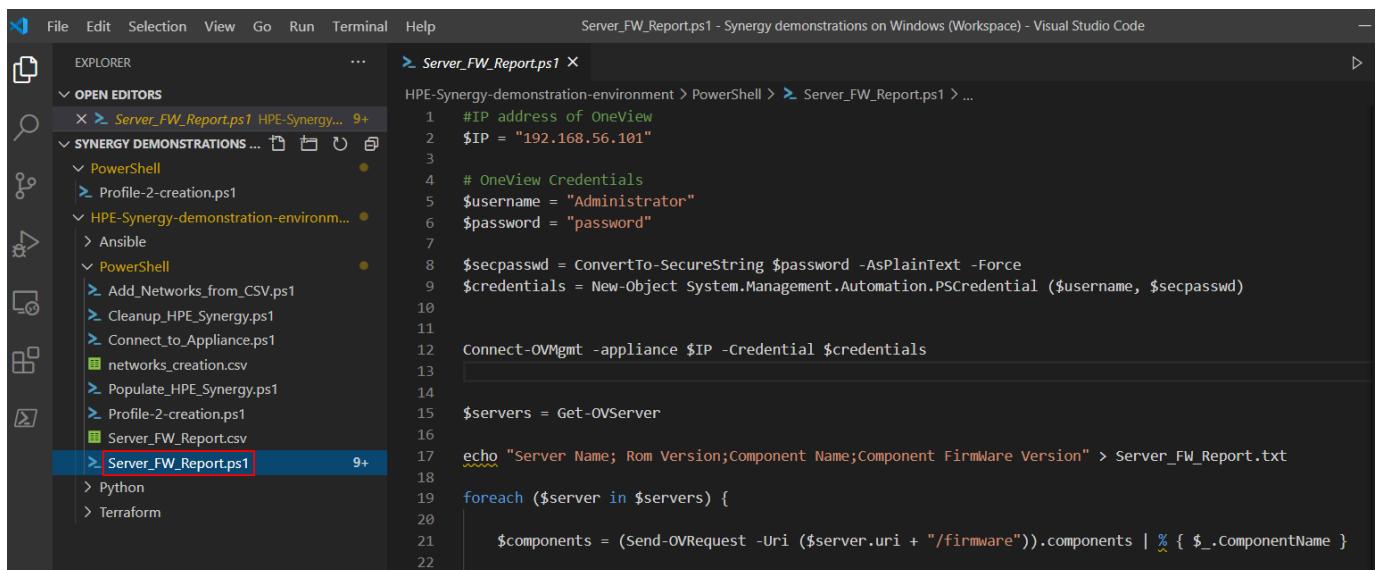
## PowerShell – Scenario 2 - Creating a report

In this scenario, we are going to demonstrate how to generate a Synergy FW inventory report of all managed compute modules using the following output format:

Server Name	Rom Version	Component Name	Component Firmware Version
Server1	P89 v2.42 (04/25/2017)	HPE Smart Storage Battery 1	Firmware 1.1
Server1	P89 v2.42 (04/25/2017)	Intelligent Platform Abstraction	Data 25.05
Server1	P89 v2.42 (04/25/2017)	Smart Array P440ar Controller	2.40

**Notice:** This scenario can be run with either the initial or final configuration snapshot.

- Open VS Code using the **Synergy demonstrations on Windows** workspace
- Open the script `Server_FW_Report.ps1` located in the **PowerShell** folder of the **HPE-Synergy-demonstration-environment** repository:



```
#IP address of OneView
$IP = "192.168.56.101"

# OneView Credentials
$username = "Administrator"
$password = "password"

$secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)

Connect-OVgmt -appliance $IP -Credential $credentials

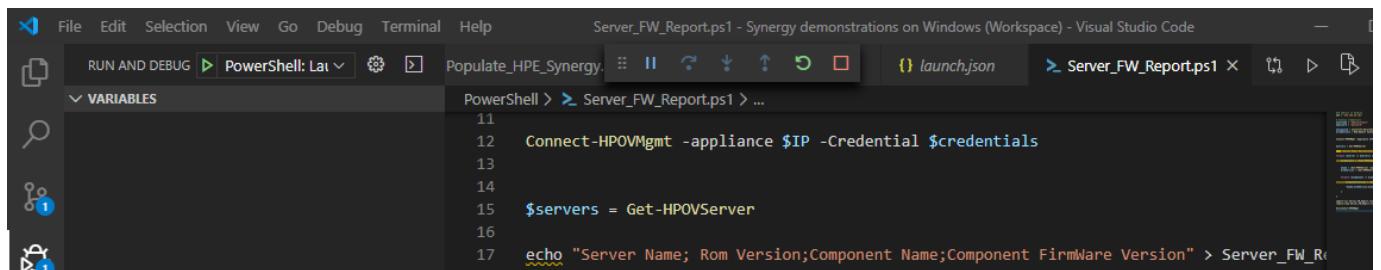
$servers = Get-OVServer

echo "Server Name; Rom Version;Component Name;Component FirmWare Version" > Server_FW_Report.txt

foreach ($server in $servers) {

    $components = (Send-OVRequest -Uri ($server.uri + "/firmware")).components | % { $_.ComponentName }
```

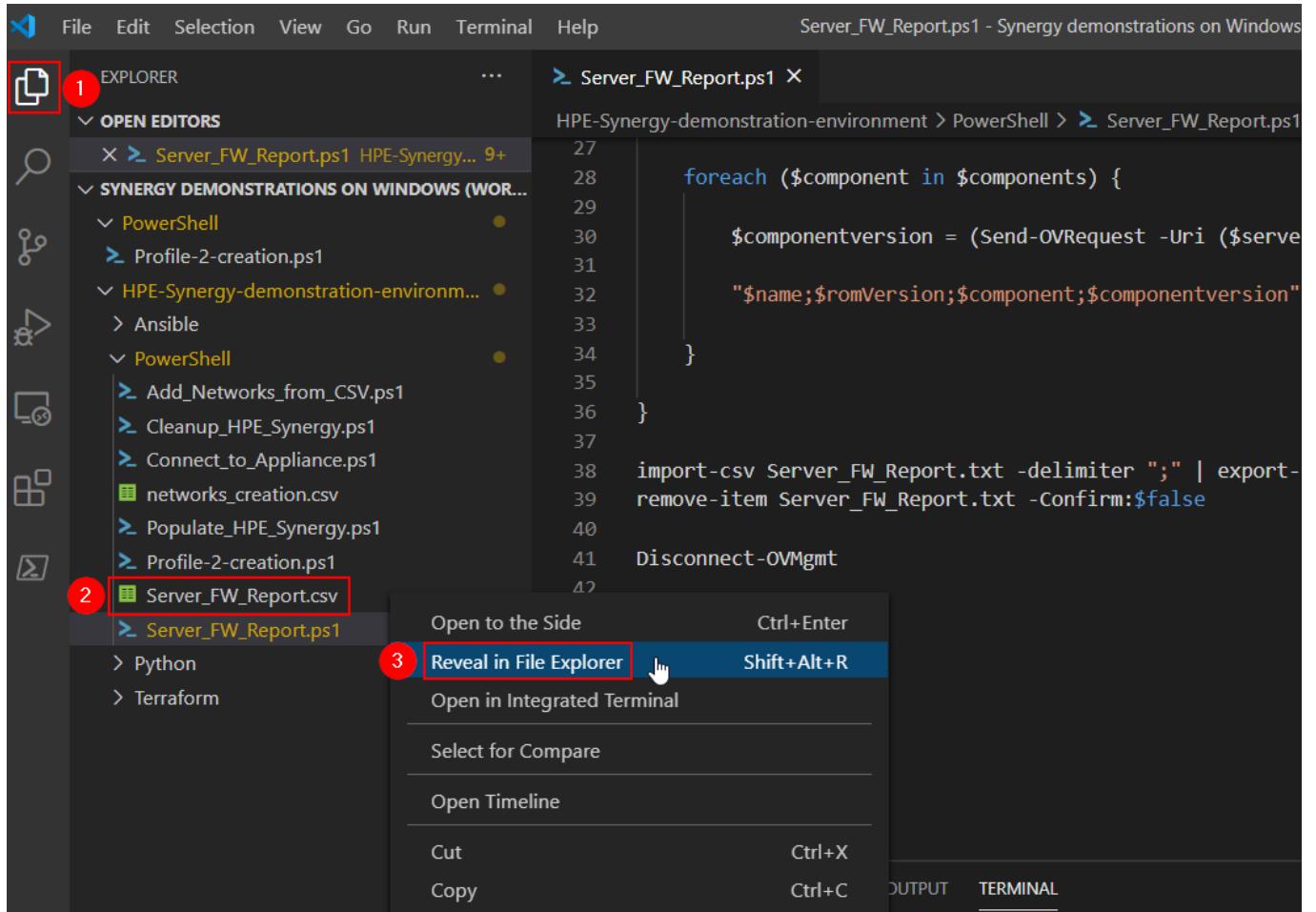
- Then press **F5** to run the script.



```
PowerShell > Server_FW_Report.ps1 > ...
11
12 Connect-HPOVgmt -appliance $IP -Credential $credentials
13
14
15 $servers = Get-HPOVServer
16
17 echo "Server Name; Rom Version;Component Name;Component FirmWare Version" > Server_FW_Report.txt
```

The script generates a CSV file `Server_FW_Report.csv` in the same folder as where the script is located.

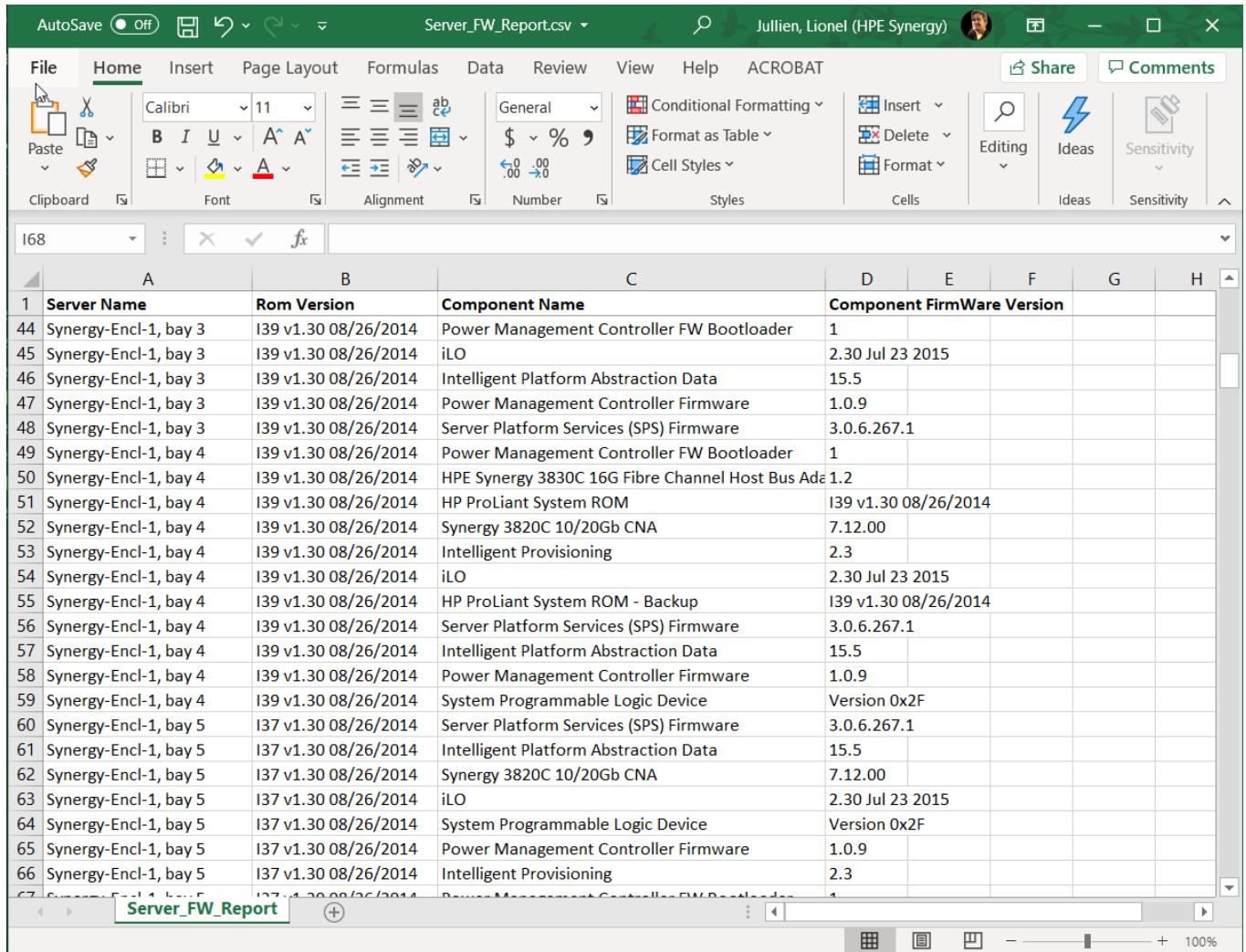
- Go back to the **Explorer**, right-click on `Server_FW_Report.csv` and select **Reveal in File Explorer**



This will open the Windows explorer

HPE-Synthetic-demonstration-environment > PowerShell				
	Name	Date modified	Type	Size
Hewlett Packard	Add_Networks_from_CSV.ps1	09/07/20 12:03 PM	PS1 File	4 KB
Personal	Cleanup_HPE_Synthetic.ps1	09/06/20 4:01 PM	PS1 File	9 KB
	Connect_to_Appliance.ps1	09/07/20 11:19 AM	PS1 File	1 KB
	networks_creation.csv	03/06/20 11:39 AM	Microsoft Excel Co...	1 KB
	Populate_HPE_Synthetic.ps1	09/06/20 7:23 PM	PS1 File	38 KB
	Profile-2-creation.ps1	09/07/20 11:59 AM	PS1 File	4 KB
	Server_FW_Report.csv	09/07/20 12:21 PM	Microsoft Excel Co...	61 KB
	Server_FW_Report.ps1	09/07/20 12:19 PM	PS1 File	2 KB

- Open the CSV file in Excel to show the generated Synergy Firmware inventory report of all managed compute modules. You might need to re-arrange the columns to get a nice presentation as illustrated below:



The screenshot shows a Microsoft Excel spreadsheet titled "Server\_FW\_Report.csv". The table has the following structure:

	A	B	C	D	E	F	G	H
1	<b>Server Name</b>	<b>Rom Version</b>	<b>Component Name</b>	<b>Component FirmWare Version</b>				
44	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Power Management Controller FW Bootloader	1				
45	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	iLO	2.30	Jul 23 2015			
46	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Intelligent Platform Abstraction Data	15.5				
47	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Power Management Controller Firmware	1.0.9				
48	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Server Platform Services (SPS) Firmware	3.0.6.267.1				
49	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Power Management Controller FW Bootloader	1				
50	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HPE Synergy 3830C 16G Fibre Channel Host Bus Adapter	1.2				
51	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HP ProLiant System ROM	I39 v1.30 08/26/2014				
52	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Synergy 3820C 10/20Gb CNA	7.12.00				
53	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Intelligent Provisioning	2.3				
54	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	iLO	2.30	Jul 23 2015			
55	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HP ProLiant System ROM - Backup	I39 v1.30 08/26/2014				
56	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Server Platform Services (SPS) Firmware	3.0.6.267.1				
57	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Intelligent Platform Abstraction Data	15.5				
58	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Power Management Controller Firmware	1.0.9				
59	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	System Programmable Logic Device	Version 0x2F				
60	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Server Platform Services (SPS) Firmware	3.0.6.267.1				
61	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Intelligent Platform Abstraction Data	15.5				
62	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Synergy 3820C 10/20Gb CNA	7.12.00				
63	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	iLO	2.30	Jul 23 2015			
64	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	System Programmable Logic Device	Version 0x2F				
65	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Power Management Controller Firmware	1.0.9				
66	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Intelligent Provisioning	2.3				

**Note:** This report may contain some null component names because we are using a simulated environment.

This concludes PowerShell – Scenario 2

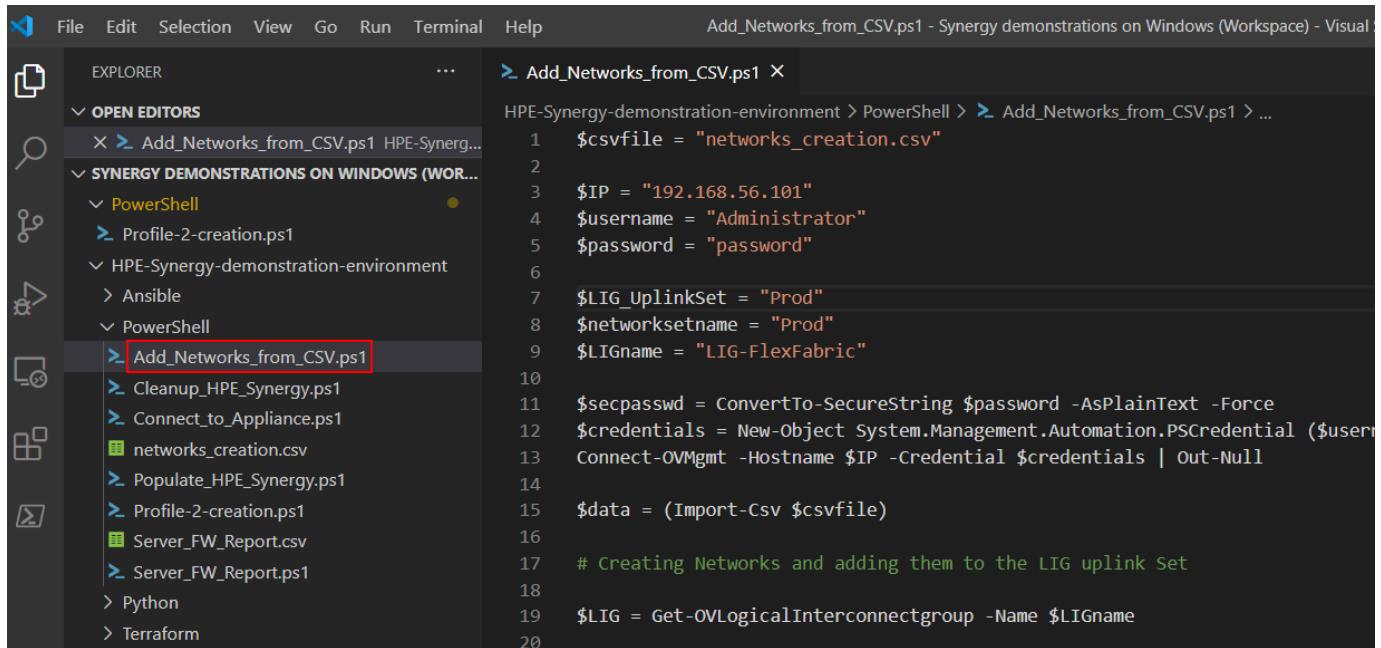


## PowerShell – Scenario 3 - Accelerating a configuration change

In this scenario, we are going to create new networks in OneView from an Excel spreadsheet (CSV file) and assign all these networks to any of the existing Server Profiles using the network set *Prod*.

**Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a configured Logical Enclosure is available)

- Open VS Code using the **Synergy demonstrations on Windows** workspace
- Open the script `Add_Network_from_CSV.ps1` located in the **PowerShell** folder of the **HPE-Synergy-demonstration-environment** repository:



The screenshot shows the Visual Studio Code interface with the following details:

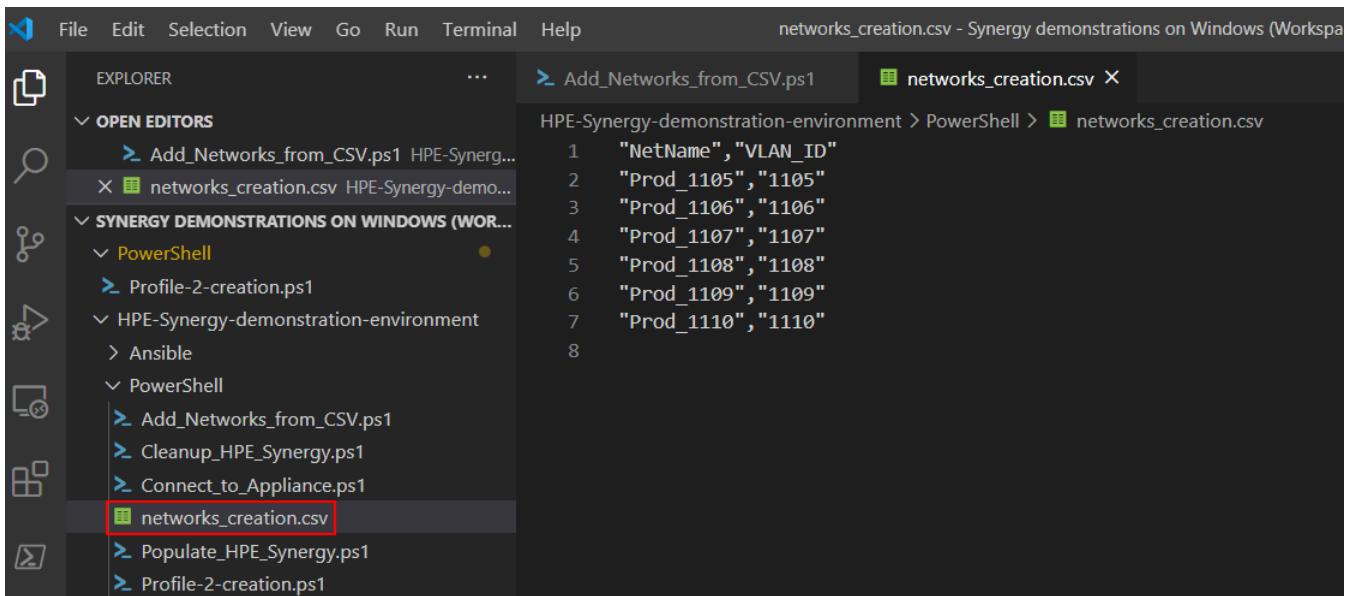
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Status Bar:** Add\_Networks\_from\_CSV.ps1 - Synergy demonstrations on Windows (Workspace) - Visual Studio Code
- Explorer View:** Shows the project structure:
  - OPEN EDITORS: Add\_Networks\_from\_CSV.ps1 (highlighted)
  - SYNERGY DEMONSTRATIONS ON WINDOWS (WORKSPACE):
    - PowerShell
      - Profile-2-creation.ps1
      - HPE-Synergy-demonstration-environment
        - Ansible
        - PowerShell
          - Add\_Networks\_from\_CSV.ps1 (highlighted)
          - Cleanup\_HPE\_Synergy.ps1
          - Connect\_to\_Appliance.ps1
          - networks\_creation.csv
          - Populate\_HPE\_Synergy.ps1
          - Profile-2-creation.ps1
          - Server\_FW\_Report.csv
          - Server\_FW\_Report.ps1
        - Python
        - Terraform
  - Editor View:** Displays the content of the `Add_Networks_from_CSV.ps1` script:

```
$csvfile = "networks_creation.csv"
$IP = "192.168.56.101"
$username = "Administrator"
$password = "password"
$LIG_UplinkSet = "Prod"
$networksetname = "Prod"
$LIGname = "LIG-FlexFabric"

$secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
Connect-OVMgmt -Hostname $IP -Credential $credentials | Out-Null

$data = (Import-Csv $csvfile)
# Creating Networks and adding them to the LIG uplink Set
$LIG = Get-OVLogicalInterconnectgroup -Name $LIGname
```

- Notice that in the folder, there is also a CSV file, `networks_creation.csv`. This file contains a list of networks with their VLAN IDs that the script will use to create the Ethernet Networks, i.e. **Prod\_1105**, **Prod\_1106**, ... **Prod\_1110** in our Synergy environment.



**Note:** The following script can be used to automatically create the CSV file:

```

Add-Content networks_creation.csv -value '"NetName","VLAN_ID"'

$networks = @(
    "Prod_1105", "1105", "Prod_1106", "1106", "Prod_1107", "1107", "Prod_1108", "1108", "Prod_1109", "1109",
    "Prod_1110", "1110" )

$networks | foreach { Add-Content -Path networks_creation.csv -Value $_ }

```

This script generates the file `networks_creation.csv` and is built as follows:

A1	:	X	✓	fx	Ne
	A	B	C		
1	NetName	VLAN_ID			
2	Prod_1105	1105			
3	Prod_1106	1106			
4	Prod_1107	1107			
5	Prod_1108	1108			
6	Prod_1109	1109			
7	Prod_1110	1110			
8					
9					

- Press **F5** to run the script and start the creation of the six new Ethernet networks.
- The first step of the script is the creation of the six networks imported from the CSV file in OneView:

```

DEBUG CONSOLE PROBLEMS 1 OUTPUT TERMINAL

Adding Network: Prod_1105 to Uplink Set: Prod

Update from group LE-Synergy-Local-LIG-FlexFabric
Uplink set changed.ct module state
[oooooooooooooo
[oooooooooooooooooooo

Creating Network: Prod_1108
Adding Network: Prod_1108 to Uplink Set: Prod

Creating Network: Prod_1109
Adding Network: Prod_1109 to Uplink Set: Prod

Creating Network: Prod_1110
Adding Network: Prod_1110 to Uplink Set: Prod

Updating all Logical Interconnects from the Logical Interconnect Group: LIG-FlexFabric

Please wait...

```

- In OneView **Networks** page, show the progression of the 6 networks creation:

Name	VLAN	Type
Deployment	1500	Ethernet
ESX Mgmt	1131	Ethernet
ESX vMotion	1132	Ethernet
Mgmt	100	Ethernet
Prod_1101	1101	Ethernet
Prod_1102	1102	Ethernet
Prod_1103	1103	Ethernet
Prod_1104	1104	Ethernet
Prod_1105	1105	Ethernet
Prod_1106	1106	Ethernet
Prod_1107	1107	Ethernet
Prod_1108	1108	Ethernet
Prod_1109	1109	Ethernet
Prod_1110	1110	Ethernet
SAN A FC		FC

**Deployment** Overview

General	
Type	Ethernet
VLAN	1500
Associated with IPv4 subnet ID	10.11.0
Associated with IPv6 subnet ID	none
Purpose	General
Preferred bandwidth	2.5 Gb/s
Maximum bandwidth	20 Gb/s
Smart link	Yes
Private network	No
Uplink set	<a href="#">US-Image Streamer</a>
Used by	<a href="#">1 deployment cluster</a>
Member of	no network sets

- In the Logical Interconnect Group, show that *LIG-FlexFabric* is now defined with 6 new networks in the *Prod* uplink set:

- Back to VS Code, the next step is the update of our Logical Interconnect from its Logical Interconnect Group (*LIG-FlexFabric*):

```

Creating Network: Prod_1108
Adding Network: Prod_1108 to Uplink Set: Prod

Creating Network: Prod_1109
Adding Network: Prod_1109 to Uplink Set: Prod

Creating Network: Prod_1110
Adding Network: Prod_1110 to Uplink Set: Prod

Updating all Logical Interconnects from the Logical Interconnect Group: LIG-FlexFabric
Please wait...

```

- This step takes some time (4-5mn) so take the opportunity to explain the concept of LI/LIG, show that the LI is inconsistent with the logical Interconnect group (shown in Activity):

- After a few minutes, the LI turns Green and *Prod* shows 10 networks:

The screenshot shows the HPE OneView interface. On the left, the 'Logical Interconnects' list includes 'LE-Synergy-Local-LIG-FlexFabric' (highlighted in green). In the main panel, under 'Logical Interconnect', the 'Prod' section shows '10 networks' (highlighted in yellow) and '2 uplink ports'. Below this, a detailed diagram of 'Synergy-Encl-1' shows a 4x4 grid of ports (L1-L2, 1-4) connected to four QoS queues (Q1-Q4). A callout box highlights 'Synergy-Encl-1, interconnect 3' with the state 'Configured' and expected/actual module information.

- You can then show the new networks appearing in the *US-Prod* uplink Set:

The screenshot shows the HPE OneView interface. The 'Uplink Sets' tab is selected. Under 'Prod', the 'Networks (10)' section lists ten network pairs, all highlighted in yellow. A red circle with the number 4 is at the top right of the main panel, and red circles with numbers 1, 2, and 3 are on the left sidebar near the 'Logical Interconnects' list.

- In the next step, the script is adding the 6 networks to the Network set *Prod*

```
Adding Network: Prod_1105 to NetworkSet: Prod
The network VLAN ID: Prod_1105 has been added successfully to all Server Profiles that are using the Network Set: Prod
Adding Network: Prod_1106 to NetworkSet: Prod
The network VLAN ID: Prod_1106 has been added successfully to all Server Profiles that are using the Network Set: Prod
Adding Network: Prod_1107 to NetworkSet: Prod
The network VLAN ID: Prod_1107 has been added successfully to all Server Profiles that are using the Network Set: Prod
Adding Network: Prod_1108 to NetworkSet: Prod
The network VLAN ID: Prod_1108 has been added successfully to all Server Profiles that are using the Network Set: Prod
Adding Network: Prod_1109 to NetworkSet: Prod
The network VLAN ID: Prod_1109 has been added successfully to all Server Profiles that are using the Network Set: Prod
Adding Network: Prod_1110 to NetworkSet: Prod
The network VLAN ID: Prod_1110 has been added successfully to all Server Profiles that are using the Network Set: Prod
PS C:\Users\Administrator.lj\Documents\DCS Appliance> █
```

- You can show in OneView, the additional networks

The screenshot shows the HPE OneView interface. On the left, there's a sidebar titled 'Network Sets 1' with a button '+ Create network set'. The main area is titled 'Prod' under 'Overview'. It shows a status message 'Update Completed 1s' by 'Administrator' on '1/29/20 9:07:11 am'. Below this, the 'General' section lists: Preferred bandwidth (2.5 Gb/s), Maximum bandwidth (20 Gb/s), Type (Regular), and Used by (2 server profiles). At the bottom, the 'Networks' section lists the following networks: Prod\_1101, Prod\_1103, Prod\_1105, Prod\_1107, Prod\_1109, Prod\_1102, Prod\_1104, Prod\_1106, Prod\_1108, and Prod\_1110.

- Explain the benefits of using network set and show that all networks are now presented to all Server Profiles as Connection 3 and 4 are connected to *Prod* network set:

The screenshot shows the HPE OneView interface. On the left, under 'Server Profiles 2', there is a green button labeled '+ Create profile'. Below it, two profiles are listed: 'Profile-1' and 'Profile-2', with 'Profile-2' currently selected and highlighted in green. On the right, under 'Profile-2', the 'Connections' tab is active. The 'Connections' table lists four connections:

ID	Name	Network	Port	Boot
1	Mgmt-1	Mgmt VLAN100	Mezzanine 3:1-a	Not bootable
2	Mgmt-2	Mgmt VLAN100	Mezzanine 3:2-a	Not bootable
3	Prod-NetworkSet-1	Prod (network set)	Mezzanine 3:1-c	Not bootable
4	Prod-NetworkSet-2	Prod (network set)	Mezzanine 3:2-c	Not bootable

This concludes PowerShell – Scenario 3



## Python – Scenario 1 - Day-to-day operation task automation

In this scenario, we are going to use Python and the OneView Python library to create a server profile.

**Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available)

- Open VS Code using the **Synergy demonstrations on WLS** workspace

Before demonstrating the creation of a server profile using Python, you can show the list of examples provided by the *OneView-Python* library.

- In Explorer, open the **oneview-python** folder and browse the **examples** folder and open **enclosure.py** as an example:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (1):** Shows the project structure under "MY-WSL-UBUNTU (WORKSPACE)". The "oneview-python" folder is highlighted with a red box and a red number 2. The "examples" folder is also highlighted with a red box and a red number 3. Inside "examples", the "enclosure.py" file is selected and highlighted with a red box and a red number 4.
- Code Editor:** Displays the content of the "enclosure.py" file. The code is a Python script for creating an enclosure. It includes imports for pprint, OneViewClient, HPOneViewException, and config\_loader. It defines a config dictionary with fields like ip, credentials (username and password), api\_version, enclosure\_group\_uri, enclosure\_hostname, enclosure\_username, and enclosure\_password.

```
Profile-3-creation.py • enclosures.py
lionel > oneview-python > examples > enclosure.py > ...
1  # -*- coding: utf-8 -*-
2  ##
3  # (C) Copyright [2019] Hewlett Packard Enterprise Development LP
4  #
5  # Licensed under the Apache License, Version 2.0 (the "License");
6  # you may not use this file except in compliance with the License.
7  # You may obtain a copy of the License at
8  #
9  #     http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 # See the License for the specific language governing permissions and
15 # limitations under the License.
16 ##
17
18 from pprint import pprint
19
20 from hpOneView.oneview_client import OneViewClient
21 from hpOneView.exceptions import HPOneViewException
22 from config_loader import try_load_from_file
23
24 # This example is compatible only for C7000 enclosures
25 config = {
26     "ip": "<oneview_ip>",
27     "credentials": {
28         "userName": "<username>",
29         "password": "<password>"
30     },
31     "api_version": "800",
32     "enclosure_group_uri": "/rest/enclosure-groups/06475bf3-084b-4874",
33     "enclosure_hostname": "",
34     "enclosure_username": "",
35     "enclosure_password": ""
36 }
37
```

Many examples are provided to help us to create our own Python script.

- Now open the script `Profile-3-creation.py` located in the **Python** folder of the **HPE-Synergy-demonstration-environment** repository:

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER View:**
  - 1: Shows the current workspace path: lionel > HPE-Synergy-demonstration-environment > Python > Profile-3-creation.py.
  - 2: Shows the `Profile-3-creation.py` file is open in the editor.
  - 3: Shows the `Python` folder contains `Network-creation.py`, `Profile-3-creation.py` (which is currently selected), and `Server_FW_Report.py`.
  - 4: Shows the `Profile-3-creation.py` file has been modified (M).
- Editor View:** The code editor displays the `Profile-3-creation.py` script. The code is as follows:

```

from hpOneView.oneview_client import OneViewClient
from pprint import pprint

config = {
    "ip": "192.168.56.101",
    "api_version": 1800,
    "credentials": {
        "userName": "Administrator",
        "password": "password"
    }
}

oneview_client = OneViewClient(config)

server_hardwares = oneview_client.server_hardware
server_profile_templates = oneview_client.server_profile_templates

myspt = server_profile_templates.get_by_name(
    'HPE Synergy 480 Gen9 with Local Boot for RHEL Template')

server = server_hardwares.get_by_name('Synergy-Encl-3, bay 5')

profile = myspt.get_new_profile()

profile['serverHardwareUri'] = server.data['uri']
profile['name'] = 'Profile-3'

oneview_client.server_profiles.create(profile)

```

A few explanations about this script:

- Import the HPOneView module with:

```
from hpOneView.oneview_client import OneViewClient
```

- Import the pprint function as well, we'll use it later to make output more readable

```
from pprint import pprint
```

- Provide the OneView information and credentials:

```
config = {
    "ip": "192.168.56.101",
    "api_version": 1800,
    "credentials": {
        "userName": "Administrator",
        "password": "password"
    }
}
```

- Make the connection with the appliance:

```
oneview_client = OneViewClient(config)

- Get the server hardware information for later use:
server_hardwares = oneview_client.server_hardware

- Get the server profile template information for later use:
server_profile_templates = oneview_client.server_profile_templates

- Pull the information of our Server Profile Template from server_profile_templates:
myspt = server_profile_templates.get_by_name(
    'HPE Synergy 480 Gen9 with Local Boot for RHEL Template')

- Pull the information of Server 'Synergy-Encl-3, bay 5' from server_profile_templates:
server = server_hardwares.get_by_name('Synergy-Encl-3, bay 5')

- Generate a new profile from our Server Profile Template:
profile = myspt.get_new_profile()

- Modify the 'serverHardwareUri' property of the generated profile object with the value of the selected server hardware uri:
profile['serverHardwareUri'] = server.data['uri']

- Set the server profile name:
profile['name'] = 'Profile-3'

- Create a new profile from our Server Profile Template:
oneview_client.server_profiles.create(profile)

- Turn the server on:
configuration = {
    "powerState": "On",
    "powerControl": "MomentaryPress"
}
server_power = server.update_power_state(configuration)
```

- Press **F5** to run the script and start the creation of the server profile **Profile-3**.

```

RUN File Edit Selection View Go Run Terminal Help Profile-3-creation.py - Synergy demonstrations on WSL (Workspace) - Visual Studio Code
Profile-3-creation.py X ...
lionel > HPE-Synergy-demonstration-environment > Python > Profile-3-creation.py ...
1 from hpOneView.oneview_client import OneViewClient
2 from pprint import pprint
3
4 config = {
5     "ip": "192.168.56.101",
6     "api_version": 1800,
7     "credentials": {
8         "userName": "Administrator",
9         "password": "password"
10    }
11 }
12 oneview_client = OneViewClient(config)
13
14 server_hardwares = oneview_client.server_hardware
15
16 server_profile_templates = oneview_client.server_profile_templates
17
18 myspt = server_profile_templates.get_by_name(
19     'HPE Synergy 480 Gen9 with Local Boot for RHEL Template')
20
21 server = server_hardwares.get_by_name('Synergy-Encl-3, bay 5')
22
23 profile = myspt.get_new_profile()
24
25 profile['serverHardwareUri'] = server.data['uri']
26
27 profile['name'] = 'Profile-3'
28
29
30 oneview_client.server_profiles.create(profile)
31

```

CALL STACK PAUSED ON PAUSE  
<module> Profile-3-creation.py 30:1

BREAKPOINTS  
■ Raised Exceptions  
☒ Uncaught Exceptions

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```

lionel@MIC2FZV4RN:~/oneview-python$ cd /home/lionel ; env /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python/python/bugpy/launcher 57128 -- /home/lionel/HPE-Synergy-demonstration-environment/Python/Profile-3-creation.py
lionel@MIC2FZV4RN:~$ cd /home/lionel ; env /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python/python/bugpy/launcher 52984 -- /home/lionel/HPE-Synergy-demonstration-environment/Python/Profile-3-creation.py

```

Start SL: Ubuntu-18.04 5.00\* Python 3.6.9 64-bit 0 △ 0 Go 1.14.2 Python: Current File (workspace)

- Simultaneously, you can open the web address <https://192.168.56.101/#/profiles> to show the creation of the new server profile:

**Server Profiles 3**

+ Create profile

Name

- Profile-1
- Profile-2
- Profile-3**

**Profile-3 General**

Creating Applying server hardware settings to Synergy-Encl-3, bay 5.

**General**

Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	<u>HPE Synergy 480 Gen9 with Local Boot for RHEL Template</u>
Server hardware	<u>Synergy-Encl-3, bay 5</u>
Server hardware type	<u>SY 480 Gen9 1</u>
Enclosure group	<u>EG-Synergy-Local</u>
Affinity	Device bay
Server power	On
Serial number (v)	VCGAXZL004
UUID (v)	6f6c1f73-686b-4773-898e-7cc1839c70fe
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgaxzl004

**OS Deployment**

- Once created, the server is turned on:

**Server Profiles 3**

+ Create profile

Name

- Profile-1
- Profile-2
- Profile-3**

**Profile-3 General**

Create Completed 4m6s

**General**

Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	<u>HPE Synergy 480 Gen9 with Local Boot for RHEL Template</u>
Server hardware	<u>Synergy-Encl-3, bay 5</u>
Server hardware type	<u>SY 480 Gen9 1</u>
Enclosure group	<u>EG-Synergy-Local</u>
Affinity	Device bay
Server power	<b>On</b>
Serial number (v)	VCGAXZL004
UUID (v)	6f6c1f73-686b-4773-898e-7cc1839c70fe
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgaxzl004

This concludes Python – Scenario 1

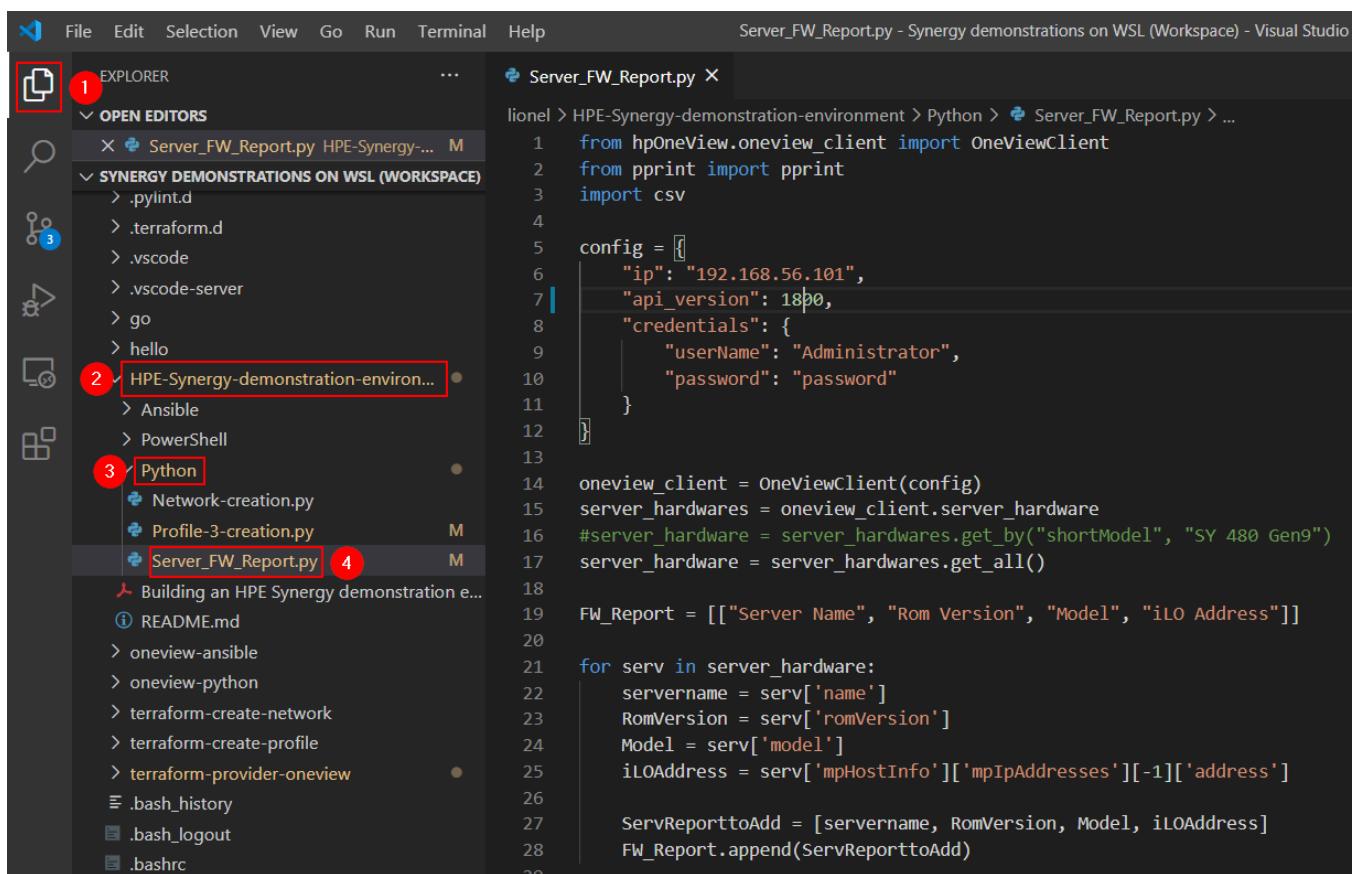
## Python – Scenario 2 – Creating a report

In this scenario, we are going to demonstrate how to generate a Synergy FW inventory report of all managed compute modules using the following output format:

Server Name	Rom Version	Model	iLO Address
Synergy-Encl-3, bay 5	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.19
Synergy-Encl-3, bay 8	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.22
Synergy-Encl-3, bay 7	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.21
Synergy-Encl-1, bay 7	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.6

**Import notice:** This scenario can be run with either the initial or final configuration snapshot.

- Open VS Code using the **Synergy demonstrations on WLS** workspace.
- Now open the script `Server_FW_Report.py` located in the **Python** folder of the **HPE-Synergy-demonstration-environment** repository:



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Title Bar:** Server\_FW\_Report.py - Synergy demonstrations on WSL (Workspace) - Visual Studio
- Explorer Panel (Left):**
  - ① OPEN EDITORS: Shows Server\_FW\_Report.py (marked with a red box).
  - ② SYNERGY DEMONSTRATIONS ON WSL (WORKSPACE): Shows the workspace structure with subfolders like Ansible, PowerShell, and Python.
  - ③ Python: Shows the Python folder containing Network-creation.py, Profile-3-creation.py, and Server\_FW\_Report.py (marked with a red box).
  - ④ Server\_FW\_Report.py: The current file being edited, highlighted with a red box and a red number 4 indicating it's the active tab.
- Code Editor (Right):**

```

from hpOneView.oneview_client import OneViewClient
from pprint import pprint
import csv

config = {
    "ip": "192.168.56.101",
    "api_version": 1800,
    "credentials": {
        "userName": "Administrator",
        "password": "password"
    }
}

oneview_client = OneViewClient(config)
server_hardwares = oneview_client.server_hardware
#server_hardware = server_hardwares.get_by("shortModel", "SY 480 Gen9")
server_hardware = server_hardwares.get_all()

FW_Report = [[ "Server Name", "Rom Version", "Model", "iLO Address"]]

for serv in server_hardware:
    servername = serv[ 'name' ]
    RomVersion = serv[ 'romVersion' ]
    Model = serv[ 'model' ]
    iLOAddress = serv[ 'mpHostInfo' ][ 'mpIpAddresses' ][ -1 ][ 'address' ]

    ServReporttoAdd = [servername, RomVersion, Model, iLOAddress]
    FW_Report.append(ServReporttoAdd)

```

A few explanations about this script:

- We use the module `server_hardwares` to get the FW inventory information of each server:

```
server.hardware = server_hardwares.get_all()
```

- Then we collect the `name`, `romversion`, `model` and iLO address attributes of each server to build the report:

```
for serv in server.hardware:
    servername = serv['name']
    romVersion = serv['romVersion']
    Model = serv['model']
    iLOAddress = serv['mpHostInfo']['mpIpAddresses'][-1]['address']
```

- Then we create `FW_Report.csv` to generate the FW inventory report for all servers:

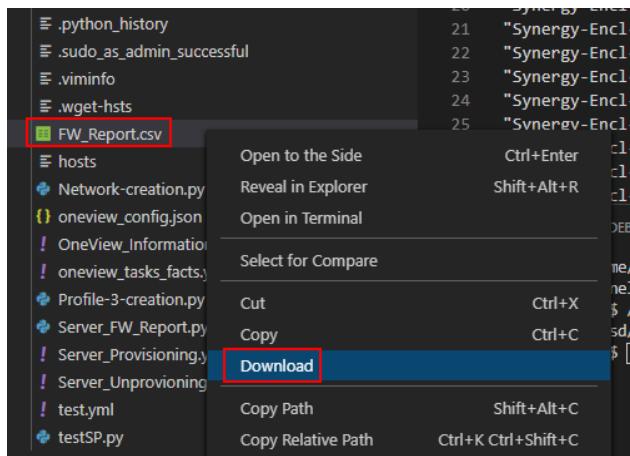
```
with open('FW_Report.csv', 'w') as file:
    writer = csv.writer(file)
    writer.writerow(FW_Report)
```

- To generate the Firmware inventory report, press **F5** and VS Code will execute the script.
- Once run, you should find a new **FW\_Report.csv** file in your user home directory:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "SYNTERGY DEMONSTRATIONS ON WSL (WORKSPACE)". Files include Network-creation.py, Profile-3-creation.py, Server\_FW\_Report.py, README.md, oneview-ansible, oneview-python, terraform-create-network, terraform-create-profile, terraform-provider-oneview, .bash\_history, .bash\_logout, .bashrc, .gitconfig, .lessht, .profile, .python\_history, .sudo\_as\_admin\_successful, .viminfo, .wget-hsts, and FW\_Report.csv.
- Code Editor:** Displays the Python script `Server_FW_Report.py`. The code uses the `OneViewClient` to get server hardware information and then writes it to a CSV file named `FW_Report.csv`.
- Terminal:** Shows the command run in the terminal: `lionel@MIC2FZV4RN:~$ cd /home/lionel ; env /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python/launcher 62881 -- /home/lionel/HPE-Synergy-demonstration-environment/Python/Server_FW_Report.py`.
- Bottom Status Bar:** Shows the environment as "WSL: Ubuntu-18.04", "Python 3.6.9 64-bit", and "Python: Current File (workspace)".

- Right-click the file and select **Download**



- Then open the file with Excel from your downloaded folder:

A	B	C	D	E
1 Server Name	Rom Version	Model	iLO Address	
2 Synergy-Encl-3, bay 4	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.18	
3 Synergy-Encl-3, bay 3	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.17	
4 Synergy-Encl-3, bay 5	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.19	
5 Synergy-Encl-3, bay 8	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.22	
6 Synergy-Encl-3, bay 7	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.21	
7 Synergy-Encl-3, bay 6	I43 v1.00 (06/20/2016)	Synergy 660 Gen10	172.18.31.6	
8 Synergy-Encl-3, bay 11	I42 v1.00 (06/20/2016)	Synergy 480 Gen10	172.18.31.5	
9 Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.4	
10 Synergy-Encl-1, bay 7	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.6	
11 Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.5	
12 Synergy-Encl-1, bay 8	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.7	
13 Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.2	
14 Synergy-Encl-1, bay 6	I43 v1.00 (06/20/2016)	Synergy 660 Gen10	172.18.31.2	
15 Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	Synergy 480 Gen10	172.18.31.1	
16 Synergy-Encl-2, bay 3	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.10	
17 Synergy-Encl-2, bay 4	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.11	
18 Synergy-Encl-2, bay 5	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.12	
19 Synergy-Encl-2, bay 6	I43 v1.00 (06/20/2016)	Synergy 660 Gen10	172.18.31.4	
20 Synergy-Encl-2, bay 7	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.14	
21 Synergy-Encl-2, bay 11	I42 v1.00 (06/20/2016)	Synergy 480 Gen10	172.18.31.3	
22 Synergy-Encl-2, bay 8	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.15	
23 Synergy-Encl-5, bay 3	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.42	
24 Synergy-Encl-5, bay 4	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.43	
25 Synergy-Encl-5, bay 9	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.38	
26 Synergy-Encl-5, bay 5	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.44	
27 Synergy-Encl-5, bay 6	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.45	
28 Synergy-Encl-5, bay 12	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.41	
29 Synergy-Encl-5, bay 11	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.40	
30 Synergy-Encl-5, bay 10	I37 v1.30 08/26/2014	HPE Synergy 480 Gen9 Compute Module	172.18.6.39	
31 Synergy-Encl-4, bay 2	I39 v1.30 08/26/2014	HPE Synergy 660 Gen9 Compute Module	172.18.6.26	

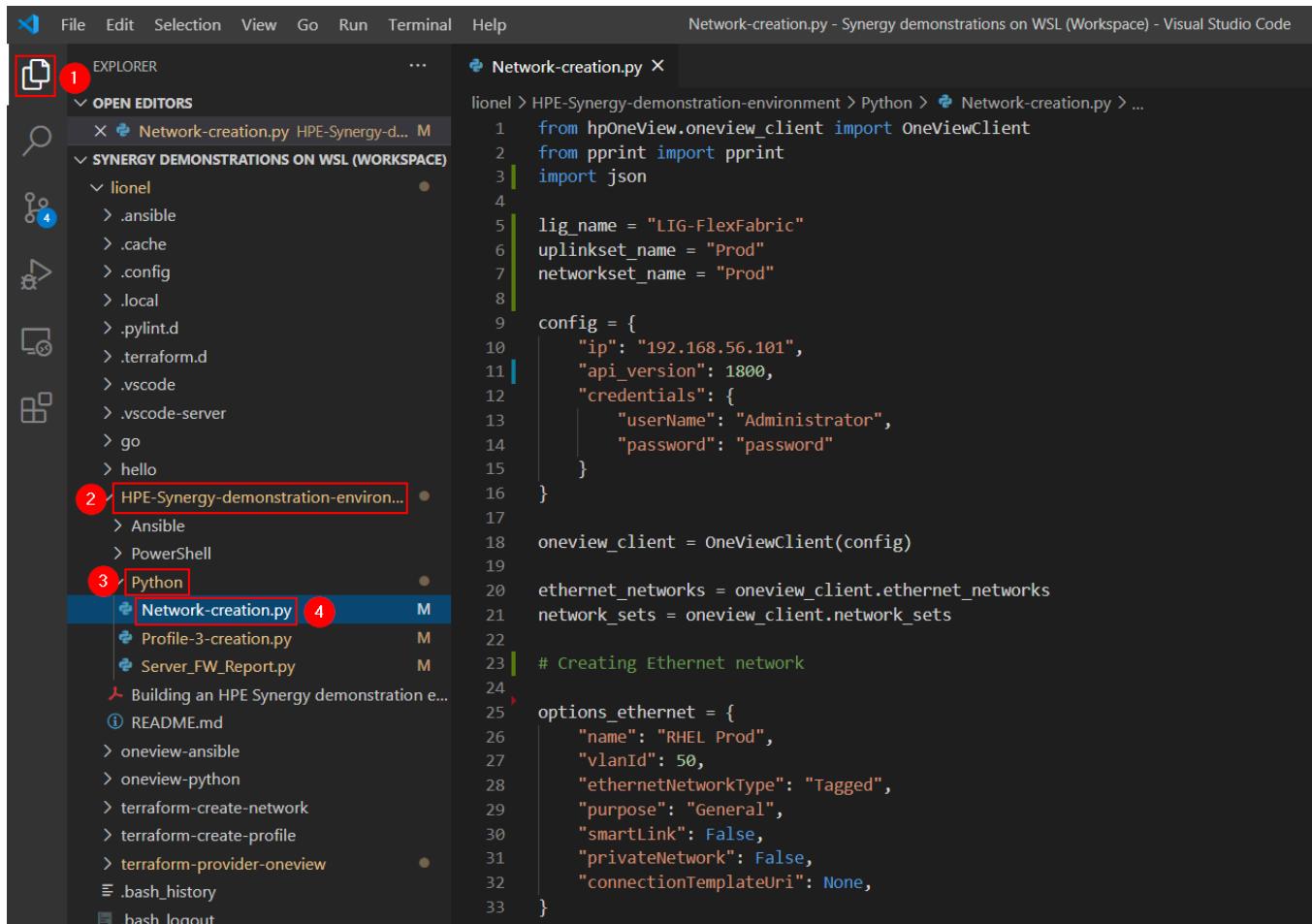
This concludes Python – Scenario 2

## Python – Scenario 3 - Accelerating a configuration change

In this scenario, we are going to use Python to create a new network in OneView (Name: *RHEL Prod*, VLANID: 50), then add this network to the uplink set *US-Prod* and to the network set *Prod* so that any of the existing Server Profiles using the network set *Prod* will be automatically connected to this new network.

**Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available)

- Open VS Code using the **Synergy demonstrations on WLS** workspace
- From the **Explorer**, open the script `Network-creation.py` located in the Python folder of the **HPE-Synergy-demonstration-environment** repository:



```

File Edit Selection View Go Run Terminal Help Network-creation.py - Synergy demonstrations on WSL (Workspace) - Visual Studio Code

1 EXPLORER
  OPEN EDITORS
    Network-creation.py HPE-Synergy-d... M
  SYNERGY DEMONSTRATIONS ON WSL (WORKSPACE)
    lionel
      .ansible
      .cache
      .config
      .local
      .pylint.d
      .terraform.d
      .vscode
      .vscode-server
      go
      hello
    HPE-Synergy-demonstration-environ...
      Ansible
      PowerShell
    Python
      Network-creation.py M
      Profile-3-creation.py M
      Server_FW_Report.py M
      Building an HPE Synergy demonstration e...
      README.md
      oneview-ansible
      oneview-python
      terraform-create-network
      terraform-create-profile
      terraform-provider-oneview
      .bash_history
      bash_logout

Network-creation.py ×
lionel > HPE-Synergy-demonstration-environment > Python > Network-creation.py > ...
1  from hpOneView.oneview_client import OneViewClient
2  from pprint import pprint
3  import json
4
5  lig_name = "LIG-FlexFabric"
6  uplinkset_name = "Prod"
7  networkset_name = "Prod"
8
9  config = {
10    "ip": "192.168.56.101",
11    "api_version": 1800,
12    "credentials": {
13      "userName": "Administrator",
14      "password": "password"
15    }
16  }
17
18  oneview_client = OneViewClient(config)
19
20  ethernet_networks = oneview_client.ethernet_networks
21  network_sets = oneview_client.network_sets
22
23  # Creating Ethernet network
24
25  options_ethernet = {
26    "name": "RHEL Prod",
27    "vlanId": 50,
28    "ethernetNetworkType": "Tagged",
29    "purpose": "General",
30    "smartLink": False,
31    "privateNetwork": False,
32    "connectionTemplateUri": None,
33  }

```

A few explanations about this script:

- First we create the new network using the `ethernet_network.create()` function with the network parameters that are provided:

```
options_ethernet = [
    "name": "RHEL_Prod",
    "vlanId": 50,
    "ethernetNetworkType": "Tagged",
    "purpose": "General",
    "smartLink": False,
    "privateNetwork": False,
    "connectionTemplateUri": None,
}

ethernet_network = ethernet_networks.create(options_ethernet))
```

- Then we get the Logical Interconnect Group data using the `logical_interconnect_groups.get_by_name()` function:

```
lig = logical_interconnect_groups.get_by_name(lig_name)
lig_response = lig.data
```

- Then we add the new network URI to the uplink set 'Prod' available in the LIG and we generate a copy of the LIG definition using the `.copy()` function:

```
for uplink in lig_response['uplinkSets']:
    if uplink['name'] == uplinkset_name:
        new_uplinkset_Prod_networkuris = uplink['networkUris'] + [ethernet_network_uri]
        uplink['networkUris'] = new_uplinkset_Prod_networkuris
lig_to_update = lig_response.copy()
```

- Then we update the LIG with the new definition using `.update()` function:

```
lig.update(lig_to_update)
```

- Then we update the LI from the LIG using the `logical_interconnect.update_compliance()` function:

```
logical_interconnect_updated = logical_interconnect.update_compliance()
```

- Finally we add the new network to the network set 'Prod' using the `network_set.update()` function:

```
network_sets = oneview_client.network_sets
network_set = network_sets.get_by_name(networkset_name)
networkset_networkUris = (network_set.data)['networkUris']
new_networkset_networkUris = networkset_networkUris + [ethernet_network_uri]
network_set_update = {'networkUris': new_networkset_networkUris}
network_set = network_set.update(network_set_update)
```

- To create the new network in OneView, press **F5** and VS Code will execute the script.
- Simultaneously, you can open the OneView web interface to see the progress of the script, open the **Networks** page to show the new **RHEL Prod** network:

The screenshot shows the HPE OneView web interface. On the left, the 'Networks' page is displayed with a table of existing networks. A red box labeled '1' highlights the 'Networks' tab. A red box labeled '2' highlights the row for the newly created 'RHEL Prod' network, which is selected and shown in a larger detail view on the right. The detail view is titled 'RHEL Prod' and shows the 'General' configuration for this network.

Name	VLAN	Type
Deployment	1500	Ethernet
ESX Mgmt	1131	Ethernet
ESX vMotion	1132	Ethernet
Mgmt	100	Ethernet
Prod_1101	1101	Ethernet
Prod_1102	1102	Ethernet
Prod_1103	1103	Ethernet
Prod_1104	1104	Ethernet
RHEL Prod	50	Ethernet
SAN A FC		FC
SAN A FCoE	10	FCoE
SAN B FC		FC
SAN B FCoE	11	FCoE
SVCluster-1	301	Ethernet

**RHEL Prod** Overview

General	
Type	Ethernet
VLAN	50
Associated with IPv4 subnet ID	none
Associated with IPv6 subnet ID	none
Purpose	General
Preferred bandwidth	2.5 Gb/s
Maximum bandwidth	10 Gb/s
Smart link	No
Private network	No
Uplink set	none
Used by	none
Member of	no network sets

- Then open **Logical Interconnects** page, select the FlexFabric LIG, to show the update from group task running:

The screenshot shows the HPE OneView interface with the 'Logical Interconnects' page selected. A red box highlights the 'Logical Interconnects' tab in the top navigation bar, labeled with a red '1'. Another red box highlights the 'LE-Synergy-Local-LIG-FlexFabric' logical interconnect in the list, labeled with a red '2'. The main content area displays the 'Logical Interconnect' configuration for this specific LIG. It includes sections for 'Internal' (no networks), 'Mgmt' (1 network, 2 uplink ports), 'Prod' (11 networks, 2 uplink ports), 'SAN-A-FC' (1 network, 1 uplink port), and 'SAN-B-FC' (1 network, 1 uplink port). Below this, a detailed diagram of 'Synergy-Encl-1' shows a 4x4 grid of ports. The first two columns (L1, L2) are green (Configured), while the next four columns (Q1, Q2, Q3, Q4) are grey (Not Configured). A callout box provides details about the interconnect: State: Configured, Expected: Virtual Connect SE 40Gb F8 Module for Synergy, and Actual: Virtual Connect SE 40Gb F8 Module for Synergy.

The LI is getting updated from the LIG new definition as we have added a new network.

- Then go to the **Uplink Sets** section to show the new **RHEL Prod** network in the **Prod** uplink set:

The screenshot shows the HPE OneView interface under the 'Logical Interconnects' section. A logical interconnect named 'LE-Synergy-Local-LIG-FlexFabric' is selected. The 'Uplink Sets' dropdown menu is open, highlighted with a red box and a circled '1'. The 'Prod' uplink set is expanded, indicated by a circled '2'. The configuration details for the 'Prod' uplink set are listed:

- Connection mode: Automatic
- LACP timer: Short (1s)
- LACP load balancing: Source & Destination MAC Address
- LACP failover trigger: All active uplinks transition to offline
- LACP distribute uplink ports: Disabled
- Native network: none

Below the configuration, there is a table for 'Networks (11)' which includes the 'RHEL Prod' network. Under 'Network sets', it says 'No network sets'.

- And finally, browse the **Network Set** page to show that **RHEL Prod** network has been added:

The screenshot shows the HPE OneView interface under the 'Network Sets' section. A network set named 'Prod' is selected. The 'Overview' tab is active, highlighted with a red box. The 'Prod' network set is shown with the following details:

- General settings:
  - Preferred bandwidth: 2.5 Gb/s
  - Maximum bandwidth: 20 Gb/s
  - Type: Regular
  - Used by: none
- Networks:
  - Prod\_1101 1101
  - Prod\_1102 1102
  - Prod\_1103 1103
  - Prod\_1104 1104
  - Prod\_1105 1105
  - Prod\_1106 1106
  - Prod\_1107 1107
  - Prod\_1108 1108
  - Prod\_1109 1109
  - RHEL Prod 50**

- In the console, the following is displayed:

```
lionel@MIC2FZV4RN:~$ cd /home/lionel ; env /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python.python-2020.8.106424/pythonFiles/lib/python/debugpy/launcher 63460 -- /home/lionel/HPE-Synergy-demonstration-environment/Python/Network-creation.py
Created ethernet networks 'RHEL Prod' successfully.
uri = '/rest/ethernet-networks/26272ff-6711-4548-8cf3-d132c57886f4'
Updating logical interconnect group 'LIG-FlexFabric'
Number of networks configured in the uplink set 'Prod' is now 11
Return the logical interconnect to a consistent state
Done. The current consistency state is CONSISTENT.
Updated network set 'Prod' successfully!

lionel@MIC2FZV4RN:~$
```

Ubuntu-18.04 5.00\* Python 3.6.9 64-bit X 0 △ 0 Go 1.14.2 Python: Current File (workspace) Ln 39, Col 1 Spaces: 4 UTF-8 LF Python ⌂ ⌂

The benefit of using network sets in Server Profile is that now *RHEL Prod* is now presented to all Server Profiles using the *Prod* network set.

This concludes Python – Scenario 3

## Ansible – Scenario 1 – Collecting facts in OneView

Hewlett Packard Enterprise has teamed up with several industry-leading configuration management providers, including Ansible by Red Hat®. The Ansible tool gives developers fast, scalable, and flexible automation of application configuration, deployment, and orchestration.

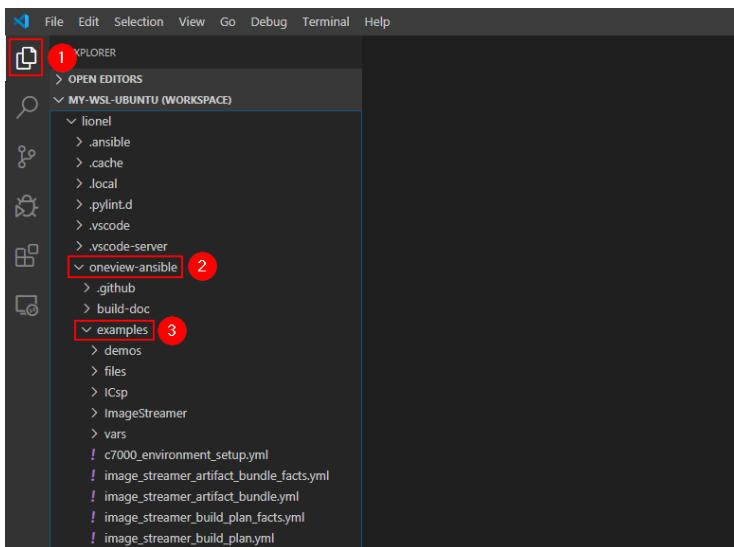
The integration of Ansible with HPE OneView extends the ability to automate the provisioning of bare-metal resources, including servers, storage, and networking. This accelerates time to value through automated, consistent provisioning.

**Note:** To learn more about Ansible with HPE OneView, see [Accelerating DevOps with HPE OneView and Ansible](#)

In this scenario, we are going to use Ansible to collect some information from our Synergy Composer DCS appliance.

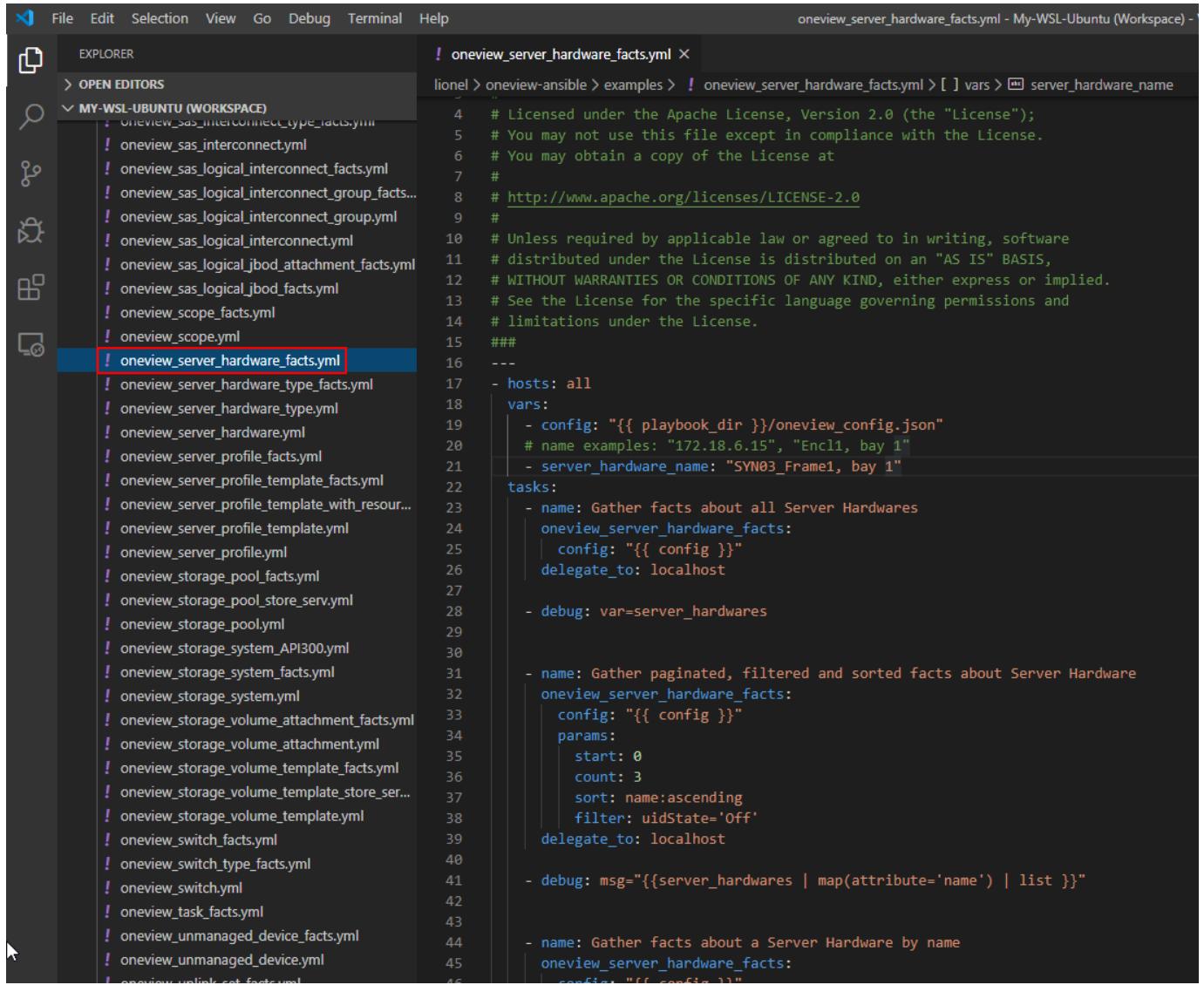
**Notice:** This scenario can be run with either the initial or final configuration snapshot.

- Open VS Code using the **Synergy demonstrations on WLS** workspace
- In Explorer, open the **oneview-ansible** folder and browse the **examples** folder:



A long list of examples is provided by the *HPE OneView-Ansible* library. Each OneView resource operation is exposed through an Ansible module.

- Open **oneview\_server\_hardware\_facts.yml** as an example:



```

File Edit Selection View Go Debug Terminal Help
oneview_server_hardware_facts.yml - My-WSL-Ubuntu (Workspace) -
EXPLORER
OPEN EDITORS
MY-WSL-UBUNTU (WORKSPACE)
! oneview_sas_interconnect_type_facts.yml
! oneview_sas_interconnect.yml
! oneview_sas_logical_interconnect_facts.yml
! oneview_sas_logical_interconnect_group_facts...
! oneview_sas_logical_interconnect_group.yml
! oneview_sas_logical_interconnect.yml
! oneview_sas_logical_jbod_attachment_facts.yml
! oneview_sas_logical_jbod_facts.yml
! oneview_scope_facts.yml
! oneview_scope.yml
! oneview_server_hardware_facts.yml
! oneview_server_hardware_type_facts.yml
! oneview_server_hardware_type.yml
! oneview_server_hardware.yml
! oneview_server_profile_facts.yml
! oneview_server_profile_template_facts.yml
! oneview_server_profile_template_with_resour...
! oneview_server_profile_template.yml
! oneview_server_profile.yml
! oneview_storage_pool_facts.yml
! oneview_storage_pool_store_serv.yml
! oneview_storage_pool.yml
! oneview_storage_system_API300.yml
! oneview_storage_system_facts.yml
! oneview_storage_system.yml
! oneview_storage_volume_attachment_facts.yml
! oneview_storage_volume_attachment.yml
! oneview_storage_volume_template_facts.yml
! oneview_storage_volume_template_store_ser...
! oneview_storage_volume_template.yml
! oneview_switch_facts.yml
! oneview_switch_type_facts.yml
! oneview_switch.yml
! oneview_task_facts.yml
! oneview_unmanaged_device_facts.yml
! oneview_unmanaged_device.yml
! oneview_uplink_set_facts.yml

! oneview_server_hardware_facts.yml ×
oneview > oneview-ansible > examples > ! oneview_server_hardware_facts.yml > [ ] vars > server_hardware_name

4 # Licensed under the Apache License, Version 2.0 (the "License");
5 # You may not use this file except in compliance with the License.
6 # You may obtain a copy of the License at
7 #
8 # http://www.apache.org/licenses/LICENSE-2.0
9 #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the License for the specific language governing permissions and
14 # limitations under the License.
15 ###
16 ---
17 - hosts: all
18 vars:
19   - config: "{{ playbook_dir }}/oneview_config.json"
20   # name examples: "172.18.6.15", "Enc11, bay 1"
21   - server_hardware_name: "SYN03_Frame1, bay 1"
22 tasks:
23   - name: Gather facts about all Server Hardwares
24     oneview_server_hardware_facts:
25       config: "{{ config }}"
26       delegate_to: localhost
27
28   - debug: var=server_hardwares
29
30
31   - name: Gather paginated, filtered and sorted facts about Server Hardware
32     oneview_server_hardware_facts:
33       config: "{{ config }}"
34       params:
35         start: 0
36         count: 3
37         sort: name:ascending
38         filter: uidState='Off'
39       delegate_to: localhost
40
41   - debug: msg="{{server_hardwares | map(attribute='name') | list }}"
42
43
44   - name: Gather facts about a Server Hardware by name
45     oneview_server_hardware_facts:
46       config: "{{ config }}"

```

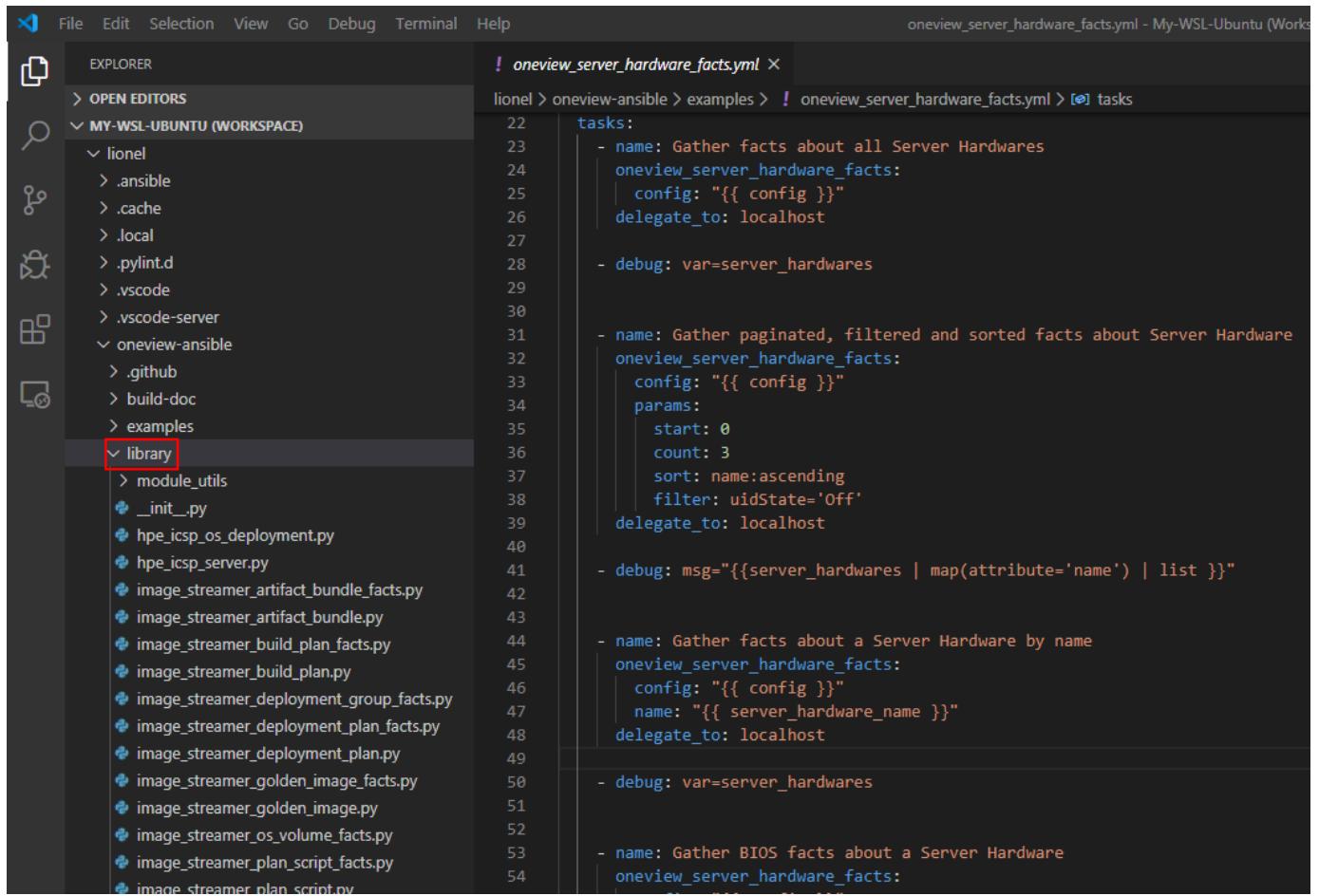
- This playbook provides several tasks examples on how to collect server hardware information. Each task uses the same `oneview_server_hardware_facts` Ansible module with different filter, options, and count parameters.

```
! oneview_server_hardware_facts.yml ×
lionel > oneview-ansible > examples > ! oneview_server_hardware_facts.yml > vars > server_hardware_name
14  # limitations under the License.
15  ###
16  ---
17  - hosts: all
18  vars:
19    - config: "{{ playbook_dir }}/oneview_config.json"
20    # name examples: "172.18.6.15", "Encl1, bay 1"
21    - server_hardware_name: "SYN03_Frame1, bay 1"
22  tasks:
23    - name: Gather facts about all Server Hardwares
24      oneview_server_hardware_facts:
25        config: "{{ config }}"
26        delegate_to: localhost
27
28    - debug: var=server_hardwares
29
30
31    - name: Gather paginated, filtered and sorted facts about Server Hardware
32      oneview_server_hardware_facts:
33        config: "{{ config }}"
34        params:
35          start: 0
36          count: 3
37          sort: name:ascending
38          filter: uidState='Off'
39        delegate_to: localhost
40
41    - debug: msg="{{server_hardwares | map(attribute='name') | list }}"
42
43
44    - name: Gather facts about a Server Hardware by name
45      oneview_server_hardware_facts:
46        config: "{{ config }}"
47        name: "{{ server_hardware_name }}"
48        delegate_to: localhost
49
50    - debug: var=server_hardwares
51
52
53    - name: Gather BIOS facts about a Server Hardware
54      oneview_server_hardware_facts:
55        config: "{{ config }}"
56        name: "{{ server_hardware_name }}"
57        options:
58          - bios
```

**Note:** Ansible uses YAML syntax for their playbooks because it is easy for humans to read/write.

**TIP:** YAML (a recursive acronym for "YAML Ain't Markup Language") is a human-readable data-serialization language that is sensitive to bad indentation! Notice that the properties vars and tasks are spaced 2 from the margin. This is called indenting and if you mess up with this, Ansible will throw an exception. Good news, the Ansible extension in VS Code does YAML validation so if your playbook is not correctly structured, you will see some warnings.

- Our Ansible modules are all located in `/oneview-ansible/library/`



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the workspace structure under "MY-WSL-UBUNTU (WORKSPACE)". The "library" folder is highlighted with a red box.
- Editor:** Displays the file `oneview_server_hardware_facts.yml`. The content is as follows:

```
lionel > oneview-ansible > examples > oneview_server_hardware_facts.yml > tasks
tasks:
  - name: Gather facts about all Server Hardwares
    oneview_server_hardware_facts:
      config: "{{ config }}"
      delegate_to: localhost
  - debug: var=server_hardwares

  - name: Gather paginated, filtered and sorted facts about Server Hardware
    oneview_server_hardware_facts:
      config: "{{ config }}"
      params:
        start: 0
        count: 3
        sort: name:ascending
        filter: uidState='Off'
      delegate_to: localhost
  - debug: msg="{{server_hardwares | map(attribute='name') | list }}"

  - name: Gather facts about a Server Hardware by name
    oneview_server_hardware_facts:
      config: "{{ config }}"
      name: "{{ server.hardware_name }}"
      delegate_to: localhost
  - debug: var=server_hardwares

  - name: Gather BIOS facts about a Server Hardware
    oneview_server_hardware_facts:
```

- Scroll-down and open the `oneview_server_hardware_facts.py` module:

```
❸ oneview_server_hardware_facts.py ×
lionel > oneview-ansible > library > ❸ oneview_server_hardware_facts.py > ...
      Set as interpreter
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  """
4  # Copyright (2016-2019) Hewlett Packard Enterprise Development LP
5  #
6  # Licensed under the Apache License, Version 2.0 (the "License");
7  # You may not use this file except in compliance with the License.
8  # You may obtain a copy of the License at
9  #
10 # http://www.apache.org/licenses/LICENSE-2.0
11 #
12 # Unless required by applicable law or agreed to in writing, software
13 # distributed under the License is distributed on an "AS IS" BASIS,
14 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 # See the License for the specific language governing permissions and
16 # limitations under the License.
17 """
18
19 ANSIBLE_METADATA = {'status': ['stableinterface'],
20                     'supported_by': 'community',
21                     'metadata_version': '1.1'}
22
23 DOCUMENTATION = """
24 ---
25 module: oneview_server_hardware_facts
26 short_description: Retrieve facts about the OneView Server Hardware.
27 description:
28     - Retrieve facts about the Server Hardware from OneView.
29 version_added: "2.3"
30 requirements:
31     - "python >= 2.7.9"
32     - "hpOneView >= 5.0.0"
33 author: "Gustavo Hennig (@GustavoHennig)"
34 options:
35     name:
36         description:
37             - Server Hardware name.
38         required: false
39         options:
40             description:
41                 - "List with options to gather additional facts about Server Hardware related resources.
42                 Options allowed: C(bios), C(javaRemoteConsoleUrl), C(environmentalConfig), C(iloSsoUrl), C(remoteConsoleUrl),
43                 C(utilization), C(firmware), C(firmwares) and C(physicalServerHardware)."
44             required: false
```

Our Ansible modules always document vital information about the module itself in the documentation section at the beginning of each module.

- OneView ansible modules (like `oneview_server_hardware_facts`) when executed, usually return some outputs in one or more variables. These variables are described at the end of the module documentation as illustrated below:

```
❸ oneview_server_hardware_facts.py ×
lionel > oneview-ansible > library > ❹ oneview_server_hardware_facts.py > ...
150   - debug: var=server_hardware_firmware
151   ...
152
153   RETURN = '''
154   server_hardwares:
155     description: Has all the OneView facts about the Server Hardware.
156     returned: Always, but can be null.
157     type: dict
158
159   server_hardware_bios:
160     description: Has all the facts about the Server Hardware BIOS.
161     returned: When requested, but can be null.
162     type: dict
163
164   server_hardware_env_config:
165     description: Has all the facts about the Server Hardware environmental configuration.
166     returned: When requested, but can be null.
167     type: dict
168
169   server_hardware_java_remote_console_url:
170     description: Has the facts about the Server Hardware java remote console url.
171     returned: When requested, but can be null.
172     type: dict
173
174   server_hardware_ilosso_url:
175     description: Has the facts about the Server Hardware iLO SSO url.
176     returned: When requested, but can be null.
177     type: dict
178
179   server_hardware_remote_console_url:
180     description: Has the facts about the Server Hardware remote console url.
181     returned: When requested, but can be null.
182     type: dict
183
184   server_hardware_utilization:
185     description: Has all the facts about the Server Hardware utilization.
186     returned: When requested, but can be null.
187     type: dict
188
189   server_hardware_firmware:
190     description: Has all the facts about the Server Hardware firmware.
191     returned: When requested, but can be null.
192     type: dict
193
194   server_hardware_firmwares:
```

- As we can see, this module returns numerous values: `server_hardwares`, `server_hardware_bios`, `server_hardware_env_config`, etc. These are variables you can use in playbook to run additional tasks.

In playbooks, you usually display on the console a returned value using:

```
- debug: var=server_hardwares
```

As described in the documentation, this will display all the OneView facts about the Server Hardware as a dictionary.

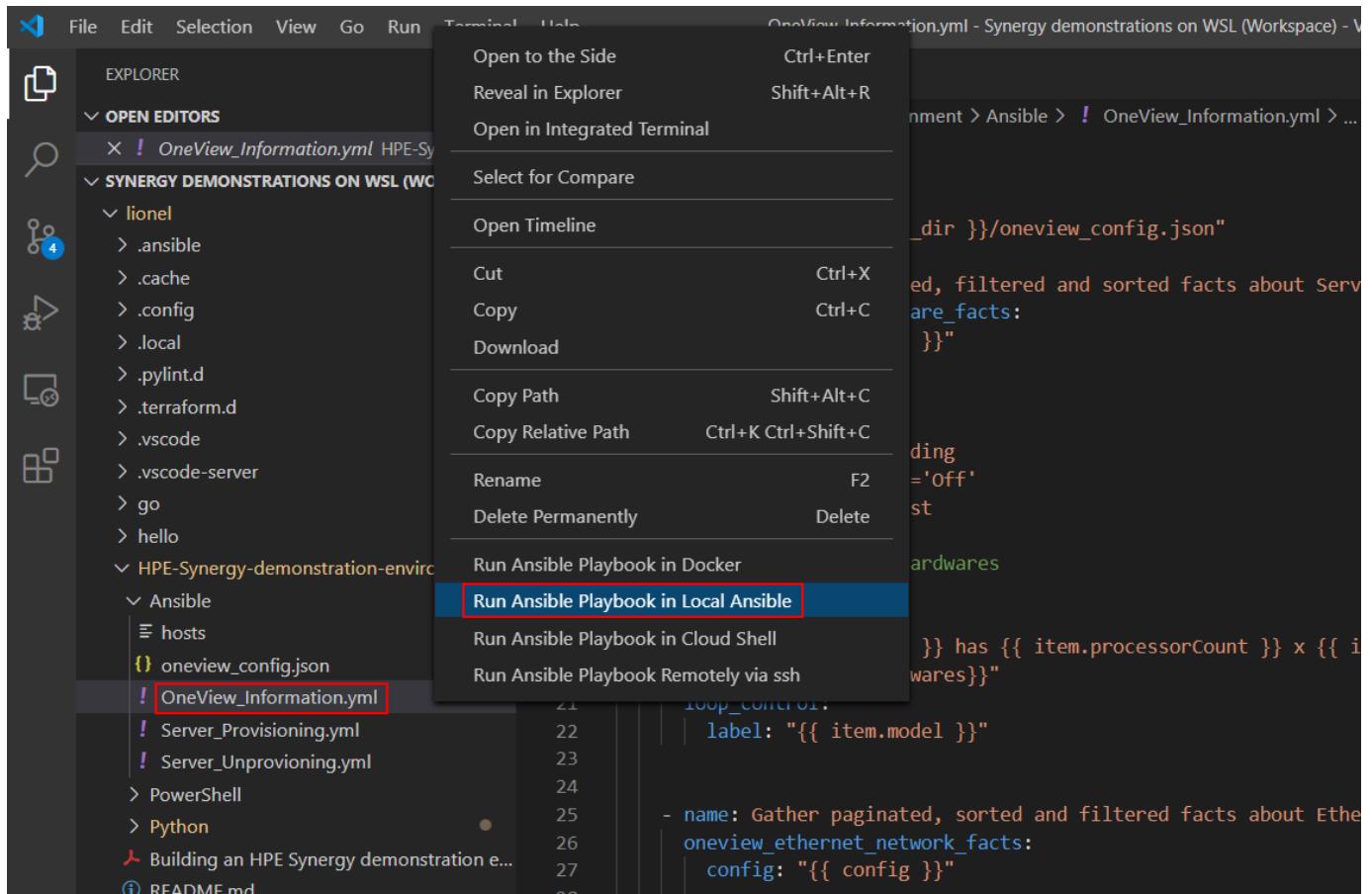
- Now open the Ansible playbook `OneView_Information.yml` located in the **Ansible** folder of the **HPE-Synergy-demonstration-environment** repository:

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** OneView\_Information.yml - Synergy demonstrations on WSL (Workspace) - Visual Studio Code.
- Explorer View (Left):**
  - 1: Open Editors (OneView\_Information.yml)
  - 2: SYNERGY DEMONSTRATIONS ON WSL (WORKSPACE) (HPE-Synergy-demonstration-environment)
  - 3: Ansible
    - hosts
    - oneview\_config.json
    - ! OneView\_Information.yml (highlighted with a red border and a red circle with the number 4)
    - ! Server\_Provisioning.yml
    - ! Server\_Unprovisioning.yml
  - PowerShell
  - Python
  - Building an HPE Synergy demonstration e...
  - README.md
- Editor View (Right):** The content of the `OneView_Information.yml` file is displayed, showing Ansible code for gathering facts about server hardware and ethernet networks.

```
---  
- hosts: localhost  
  vars:  
    - config: "{{ playbook_dir }}/oneview_config.json"  
  tasks:  
    - name: Gather paginated, filtered and sorted facts about Server Hardware  
      oneview_server_hardware_facts:  
        config: "{{ config }}"  
        params:  
          start: 0  
          count: 3  
          sort: name:ascending  
          filter: uidState='off'  
        delegate_to: localhost  
  
    #- debug: var=server_hardwares  
  
    - debug:  
        msg: "{{ item.name }} has {{ item.processorCount }} x {{ item.processorType }} processor(s)  
        loop: "{{server_hardwares}}"  
        loop_control:  
          label: "{{ item.model }}"  
  
    - name: Gather paginated, sorted and filtered facts about Ethernet Networks  
      oneview_ethernet_network_facts:  
        config: "{{ config }}"  
        params:  
          start: 0  
          count: 3  
          sort: name:ascending  
          filter: idState='off'
```

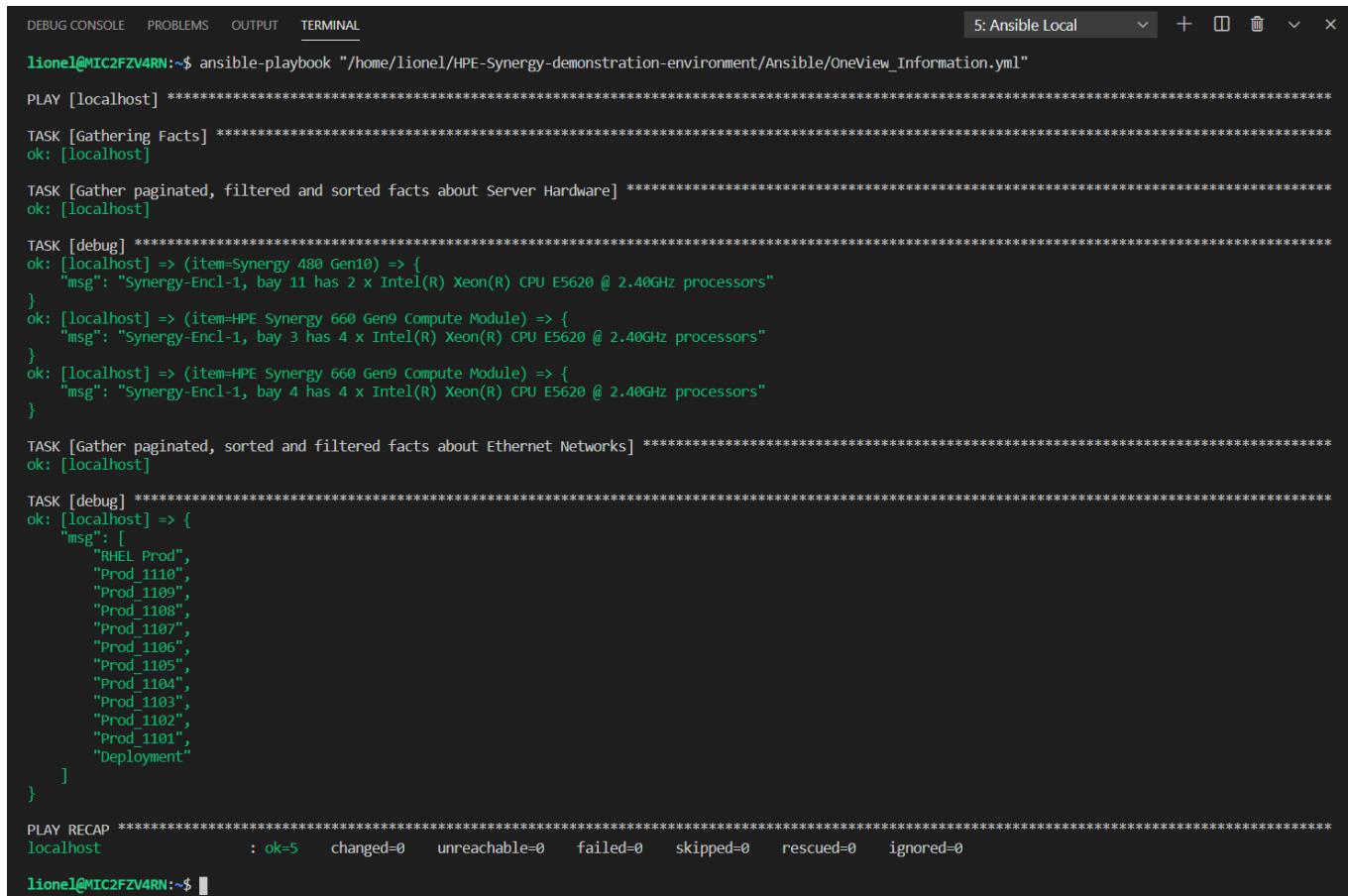
- To easily run the playbook, right-click on the file and select **Run Ansible Playbook in Local Ansible**



Notice that VS Code automatically generates and runs the command `ansible-playbook <file>` in the terminal window:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL 5: Ansible Local
export VSCODEEXT_USER_AGENT=vscooss.vscode-ansible-0.5.2
ansible-playbook "/home/lionel/HPE-Synergy-demonstration-environment/Ansible/OneView_Information.yml"
lionel@MIC2FZV4RN:~$ export VSCODEEXT_USER_AGENT=vscooss.vscode-ansible-0.5.2
lionel@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/HPE-Synergy-demonstration-environment/Ansible/OneView_Information.yml"
```

- The console outputs the following:



The screenshot shows a terminal window titled "TERMINAL" with the tab "5: Ansible Local" selected. The terminal displays the output of an Ansible playbook named "Ansible\_OneView\_Information.yml". The output includes various Ansible tasks such as "Gathering Facts", "Gather paginated, filtered and sorted facts about Server Hardware", and "Gather paginated, sorted and filtered facts about Ethernet Networks". The "Gather paginated, filtered and sorted facts about Server Hardware" task shows details for three server hardware components, including their model numbers and processor information. The "Gather paginated, sorted and filtered facts about Ethernet Networks" task shows a list of network interface names. The final "PLAY RECAP" section indicates that all hosts (localhost) were successful (ok=5, changed=0, failed=0).

```
lionel@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/HPE-Synergy-demonstration-environment/Ansible/OneView_Information.yml"
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]
TASK [Gather paginated, filtered and sorted facts about Server Hardware] ****
ok: [localhost]
TASK [debug] ****
ok: [localhost] => (item=Synergy 480 Gen10) => {
    "msg": "Synergy-Enc1-1, bay 11 has 2 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}
ok: [localhost] => (item=HPE Synergy 660 Gen9 Compute Module) => {
    "msg": "Synergy-Enc1-1, bay 3 has 4 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}
ok: [localhost] => (item=HPE Synergy 660 Gen9 Compute Module) => {
    "msg": "Synergy-Enc1-1, bay 4 has 4 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}
TASK [Gather paginated, sorted and filtered facts about Ethernet Networks] ****
ok: [localhost]
TASK [debug] ****
ok: [localhost] => {
    "msg": [
        "RHEL Prod",
        "Prod_1110",
        "Prod_1109",
        "Prod_1108",
        "Prod_1107",
        "Prod_1106",
        "Prod_1105",
        "Prod_1104",
        "Prod_1103",
        "Prod_1102",
        "Prod_1101",
        "Deployment"
    ]
}
PLAY RECAP ****
localhost : ok=5    changed=0   unreachable=0   failed=0    skipped=0   rescued=0   ignored=0
lionel@MIC2FZV4RN:~$
```

In the first Ansible debug message task, Ansible displays formatted Server hardware information. We are using the `server_hardware` variable generated by the `oneview_server_hardware_facts` module. We use the `loop` option to go through the content of `server_hardware` a dictionary object. `loop_control` is used to prevent Ansible to display the entire content of the `{{ item }}` variable but instead just the `model`.

Next In the second Ansible debug message task, Ansible displays the networks available in OneView. Networks with a general purpose are displayed using a descending sort.

With Ansible fact modules, we can gather information about almost all resources in OneView using parameters found in the module documentation to sort, filter, count, etc.

**Note:** if you are running Ubuntu 20.04, you will get a permission error when running the playbook like:

```
fatal: [localhost]: UNREACHABLE! => {"changed": false, "msg": "Failed to create temporary directory. In some cases, you may have been able to authenticate and did not have permissions on the target directory. Consider changing the remote tmp path in ansible.cfg to a path rooted in \"/tmp\", for more error information use -vvv. Failed command was: ( umask 77 && mkdir -p \"` echo /home/`/.ansible/tmp `\"&& mkdir /home/ `/.ansible/tmp/ansible-tmp-1591538856.968808-12936-59109817527365=\` echo /home/`/.ansible/tmp/ansible-tmp-1591538856.968808-12936-59109817527365 `\" ), exited with result 1, stdout output: ansible-tmp-1591538856.968808-12936-59109817527365=/home/`/.ansible/tmp/ansible-tmp-1591538856.968808-12936-59109817527365\n", "unreachable": true}
```

This is due to the known issues when running Ubuntu 20.04 on WSL 1. You can enter the following commands as a temporary workaround:

```
sudo mv /usr/bin/sleep /usr/bin/sleep.dist  
sudo ln -s /bin/true /usr/bin/sleep
```

For more information, see <https://community.spiceworks.com/topic/2275812-ubuntu-wsl-ansible-permission-error-when-running-localhost-playbook>

This concludes Ansible – Scenario 1 about collecting facts in OneView



## Ansible – Scenario 2 – Provisioning New Servers

In this scenario, we are going to use Ansible to automate the creation of two server profiles using an existing Server Profile Template.

**Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available)

One of the nice features with Ansible is that it can work against multiple systems in your infrastructure at the same time by using an inventory file, named *hosts* located in `/etc/ansible/hosts`

In the *hosts* inventory file, you can create groups of servers and use a specific group in an Ansible playbook to run some tasks that will be executed on all systems listed in this group.

To illustrate this, we are going to create a *RHEL* group in the *hosts* file with two systems, *RH-1* and *RH-2*.

But to facilitate the edition in VS Code, we are going to create our own *hosts* file located in our user home directory. To do that, we need to modify the Ansible configuration:

- From the Ubuntu console and enter:

```
sudo vi /etc/ansible/ansible.cfg
```

- Enter your password.
- Uncomment the inventory value:

```
# -----
# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

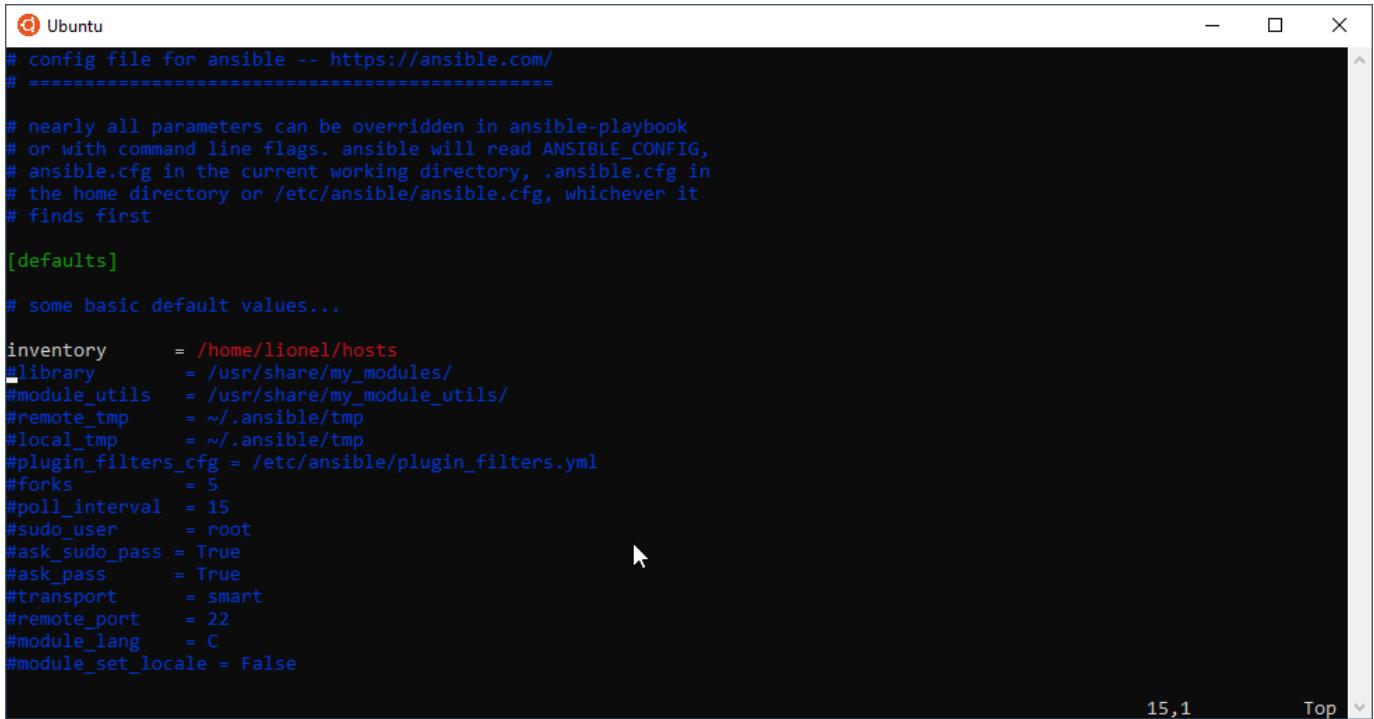
#inventory      = /etc/ansible/hosts ←
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
```

- Press the letter **i** to put the VI editor in *Insert Mode*. Then edit the inventory path with:

```
inventory      = /home/<username>/hosts
```

with <username> your username





The screenshot shows a terminal window titled "Ubuntu" with a black background and white text. It displays the contents of an Ansible configuration file. The file includes sections for inventory, library, module\_utils, remote\_tmp, local\_tmp, plugin\_filters\_cfg, forks, poll\_interval, sudo\_user, ask\_sudo\_pass, ask\_pass, transport, remote\_port, module\_lang, and module\_set\_locale. A cursor is visible in the middle of the code. At the bottom right, there are status indicators: "15,1" and "Top".

```
# config file for ansible -- https://ansible.com/
# =====

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

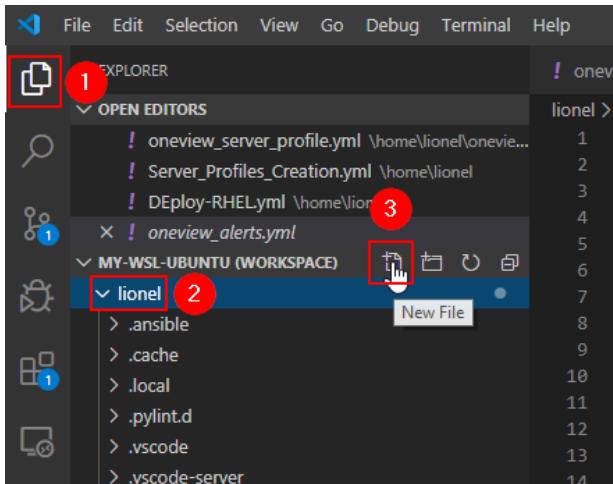
# some basic default values...

inventory      = /home/lionel/hosts
library         = /usr/share/my_modules/
module_utils    = /usr/share/my_module_utils/
remote_tmp      = ~/.ansible/tmp
local_tmp       = ~/.ansible/tmp
plugin_filters_cfg = /etc/ansible/plugin_filters.yml
forks           = 5
poll_interval   = 15
sudo_user       = root
ask_sudo_pass   = True
ask_pass         = True
transport       = smart
remote_port     = 22
module_lang     = C
module_set_locale = False
```

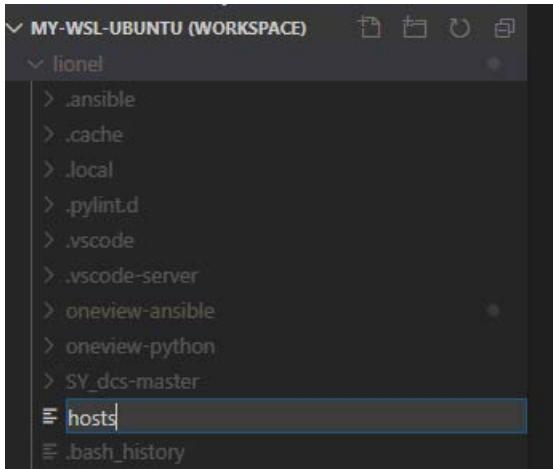
- Press **ESC** to exit *Insert Mode*, then type : (colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi

Ansible is now configured to use a *hosts* file located in our user home directory. Let's now create this file.

- Open VS Code using the **Synergy demonstrations on WLS** workspace
- in the **Explorer** view, select your home folder then click on **New File**



- Named it **hosts**:

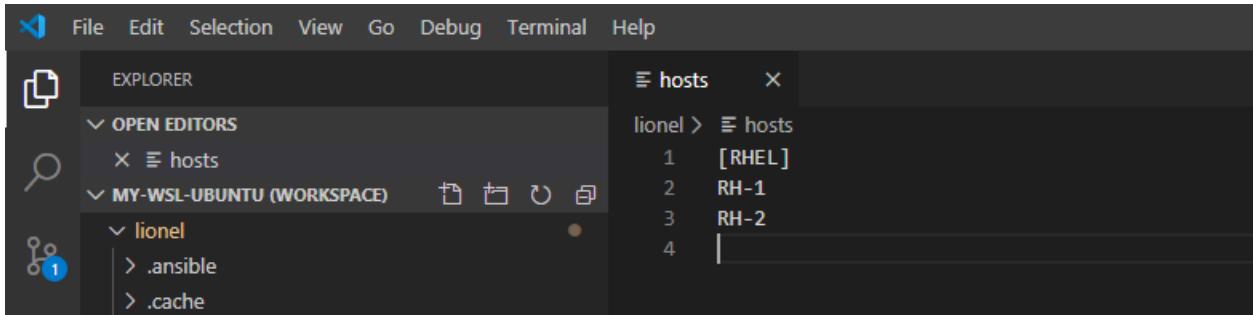


- Then add in the hosts file the following content:

```
[RHEL]
RH-1
RH-2
```

- Then save the file by pressing **CTRL + S**

**Note:** [...] defines *RHEL* as our group. *RH-1* and *RH-2* are the two systems defined in this group.



Now let's use a playbook that will use this *hosts* file to create two server profiles (RH-1 and RH-2) from an existing Server Profile Template.

- Go back to VS Code, in Explorer, open the Ansible playbook `Server_Provisioning.yml` located in the **Ansible** folder of the **HPE-Synergy-demonstration-environment** repository:

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Title Bar:** Server\_Provisioning.yml - Synergy demonstrations on WSL (Workspace) - Visual Studio Code
- Explorer View:**
  - OPEN EDITORS: Server\_Provisioning.yml (HPE-Synergy-demonstration-environment)
  - SYNTERGY DEMONSTRATIONS ON WSL (WORKSPACE):
    - lionel (4 files): .ansible, .cache, .config, .local, .pylint.d, .terraform.d, .vscode, .vscode-server, go, hello
    - HPE-Synergy-demonstration-environment (2 files): hosts, oneview\_config.json
    - Ansible (3 files): OneView\_Information.yml, Server\_Provisioning.yml (highlighted), Server\_Unprovisioning.yml
    - PowerShell, Python, Building an HPE Synergy demonstration e..., README.md
- Code Editor:** The file `Server_Provisioning.yml` is open and visible. It contains Ansible YAML code for provisioning server profiles. The code includes sections for hosts, vars, and tasks, specifically for creating server profiles from a template.

- This playbook contains two main group of tasks, one to turn off the server hardware and one to create the server profiles:

The screenshot shows the Ansible playbook `Server_Provisioning.yml` with the following tasks highlighted:

- Task 1:** `- name: Making sure the Server Hardware tied to the Server Profile Template are turned off...`
- Task 2:** `- name: Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template in OneView ...`

- In the first group of tasks, we make sure the server hardware that are using the same server hardware type as the server profile template are turned off so that we can create the server profiles. Notice we do not run these tasks using the `hosts` file but instead using `localhost`.

- 1:** We gather the facts about the server profile template we use.
- 2:** We set a variable to collect the server hardware type uri used by the server profile template we use.
- 3:** We gather the facts about the server hardware facts.
- 4:** We set a variable to collect the server hardware names that are using the server hardware type uri collected in **2**
- 5:** We turn off all server hardware using the names collected in **4**

```
! Server_Provisioning.yml •
lionel > ! Server_Provisioning.yml > name
1   ---
2   - name: Making sure the Server Hardware tied to the Server Profile Template are turned off
3     hosts: localhost
4     gather_facts: no
5     vars:
6       - config: "{{ playbook_dir }}/oneview_config.json"
7       - server_template: "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
8
9     tasks:
10    - name: Gather Facts about a Server Profile Template by name
11      oneview_server_profile_template_facts:
12        config: "{{ config }}"
13        name: "{{ server_template }}"
14        delegate_to: localhost
15
16      # - debug: var=server_profile_templates
17
18    - name: Get the Server Hardware Type Uri used by the template
19      set_fact:
20        serverHardwareTypeUri: "{{ server_profile_templates.0.serverHardwareTypeUri }}"
21
22      # - debug: var=serverHardwareTypeUri
23
24    - name: Get the Server Hardware using the SHT of the server profile template
25      oneview_server_hardware_facts:
26        config: "{{ config }}"
27        delegate_to: localhost
28
29      # - debug:
30      #   msg: "{{ server_hardwares | selectattr('serverHardwareTypeUri', 'equalto', serverHardwareTypeUri ) }}"
31
32    - set_fact:
33      server.hardware_names : "{{ server_hardwares | selectattr('serverHardwareTypeUri', 'equalto', serverHardwareTypeUri ) }}"
34
35      # - debug: var=server.hardware_names
36
37    - name: Making sure the Compute Module(s) using the SHT are turned off
38      with_items: "{{ server.hardware_names }}"
39      oneview_server_hardware:
40        config: "{{ config }}"
41        state: power_state_set
42        data:
43          name : "{{ item }}"
44          powerStateData:
45            powerState: "Off"
46            powerControl: "MomentaryPress"
47        delegate_to: localhost
48
49 > - name: Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template in OneView
```

- In the second group of tasks, we create the server profiles. Notice that this time, we run those tasks using the `hosts` file:

- ①**: We create one or more server profiles according to the *RHEL* group settings found in the hosts inventory file. The group is set by `hosts: RHEL` at the beginning of the second group of tasks.
- ②**: We defines the Server Profile Template that Ansible must use to generate the Server Profiles, the Server Profile Template is set in line 54.
- ③**: We tell Ansible to use the hosts inventory file to generate the Server Profile names.
- ④**: We display the result of the Server Profile creation task

```
---
- name: Ansible OneView Synergy playbook to deploy Compute Module(s) using a Server Profile Template
  hosts: RHEL
  gather_facts: no
  vars:
    - config: "{{ playbook_dir }}/oneview_config.json"
    - server_template: "HPE Synergy 480 Gen9 with Local Boot for RHEL Template"

  tasks:
    - name: Creating Server Profile [{{ inventory_hostname }}] from Server Profile Template [{{ server_template }}]
      oneview_server_profile: ← 1
        config: "{{ config }}"
        data:
          serverProfileTemplateName: "{{ server_template }}"
          name: "{{ inventory_hostname }}" ← 2
      delegate_to: localhost
      register: result

      #- debug: var=server_hardware

    - name: Task result of the Server Profile(s) creation
      debug:
        msg: "{{ result.msg }}" ← 3

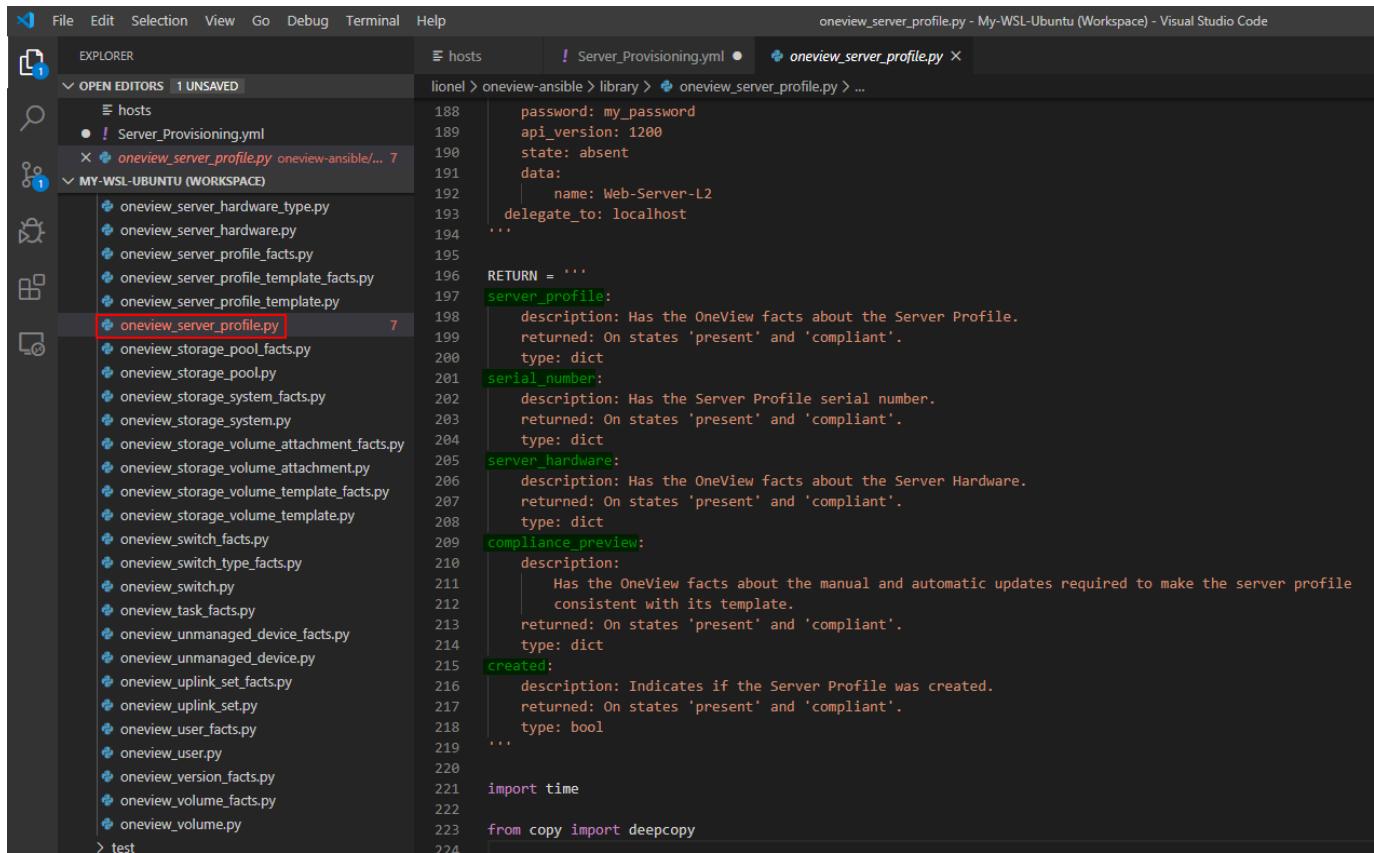
```

- Then in the next subtasks, the servers are powered-on in **①** then we display in **②** the server location.

```
- name: Powering on the Compute Module(s) [{{ server.hardware.name }}] ← 1
  oneview_server_hardware:
    config: "{{ config }}"
    state: power_state_set
    data:
      name : "{{ server.hardware.name }}"
      powerStateData:
        powerState: "On"
        powerControl: "MomentaryPress"
      delegate_to: localhost

    - debug:
        msg: "The server is located in {{ server.hardware.name }}" ← 2
```

- As described below in the `oneview_server_profile` module documentation located in `/oneview-ansible/library/oneview_server_profile.py`, `oneview_server_profile` returns 4 variables, `server_profiles`, `serial_number`, `server_hardware`, `compliance_preview` and `created`.

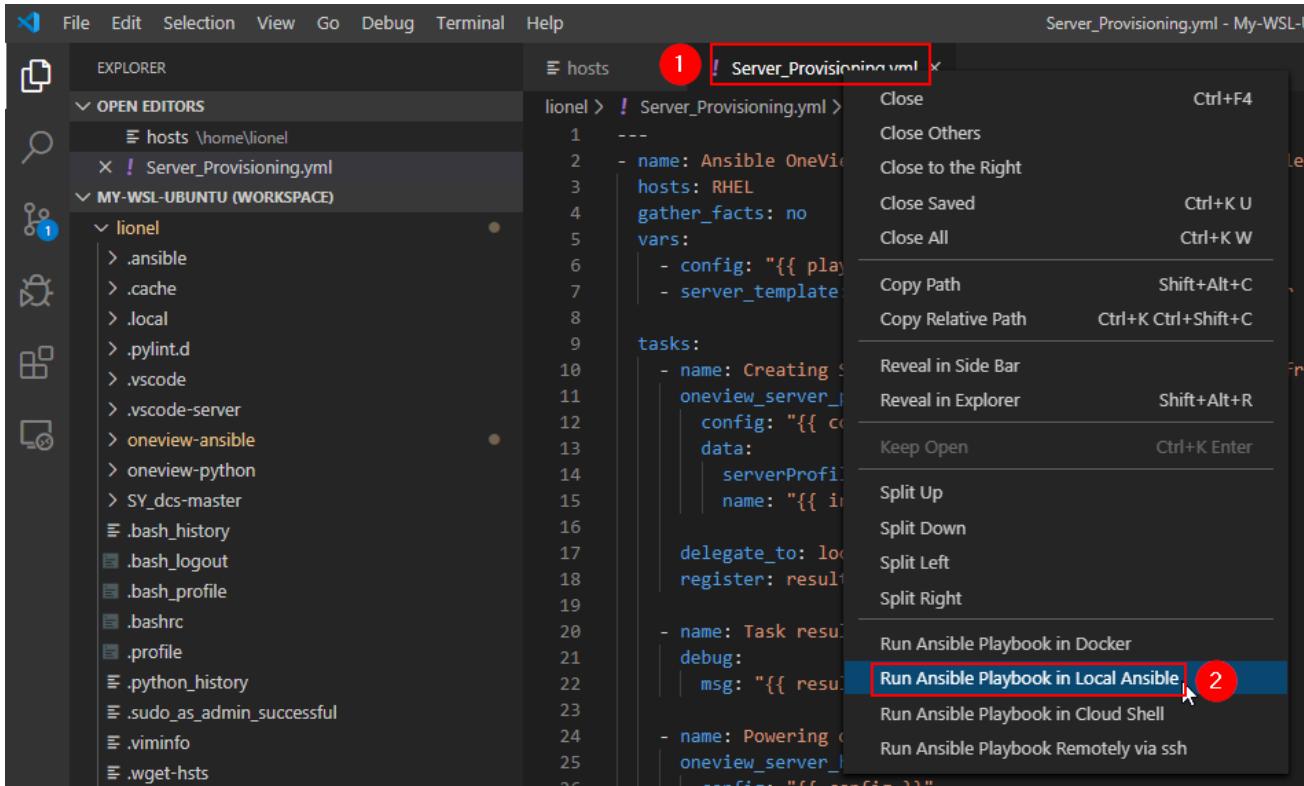


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "MY-WSL-UBUNTU (WORKSPACE)". The file `oneview_server_profile.py` is selected and highlighted.
- Code Editor:** Displays the Python code for the `oneview_server_profile` module. The code defines several variables and their descriptions. The variables are:
  - `password`: my\_password
  - `api_version`: 1200
  - `state`: absent
  - `data`: A dictionary with key `name` set to "Web-Server-L2" and key `delegate_to` set to "localhost".
  - `RETURN`: An empty string.
  - `server_profile`: A dictionary with key `description` set to "Has the OneView facts about the Server Profile.", `returned` set to "On states 'present' and 'compliant'.", and `type` set to "dict".
  - `serial_number`: A dictionary with key `description` set to "Has the Server Profile serial number.", `returned` set to "On states 'present' and 'compliant'.", and `type` set to "dict".
  - `server_hardware`: A dictionary with key `description` set to "Has the OneView facts about the Server Hardware.", `returned` set to "On states 'present' and 'compliant'.", and `type` set to "dict".
  - `compliance_preview`: A dictionary with key `description` set to "Has the OneView facts about the manual and automatic updates required to make the server profile consistent with its template.", `returned` set to "On states 'present' and 'compliant'.", and `type` set to "dict".
  - `created`: A dictionary with key `description` set to "Indicates if the Server Profile was created.", `returned` set to "On states 'present' and 'compliant'.", and `type` set to "bool".
- Bottom Status Bar:** Shows the status "160 | Page".

In our playbook, the last message we send to the console, is using one of those variables returned by the module when executed (i.e. `server_hardware`). We display only the `name` attribute using `server_hardware.name` to get only the server hardware location.

- Now to run the Ansible playbook, right-click on the tab then select **Run Ansible Playbook in Local Ansible**



- Open the OneView UI (<https://192.168.56.101/#/profiles>) to show the creation of the two server profiles:

The screenshot shows the HPE OneView interface with the 'Server Profiles' page. The 'RH-1' profile is selected and its details are displayed in the main pane. The 'Actions' button is highlighted with a red circle labeled '1'.

Name	Description
Profile-1	Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Profile-2	HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template
Profile-3	Synergy-Encl-2.bay.3
RH-1	SY 660 Gen9 1 EG-Synergy-Local
RH-2	Off VCGONC3006 fb43988d-7df3-4e6e-8619-c56ae9a32be iqn.2015-02.com.hpe:oneview-vcg0nc3006

- Open the VS Code terminal to see the outputs of our Ansible playbook tasks:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

ok: [localhost]

TASK [Get the Server Hardware Type Uri used by the template] *****
ok: [localhost]

TASK [Get the Server Hardware using the SHT of the server profile template] *****
ok: [localhost]

TASK [set_fact] *****
ok: [localhost]

TASK [Making sure the Compute Module(s) using the SHT are turned off] *****
changed: [localhost] => (item=Synergy-Enc1-3, bay 3)
changed: [localhost] => (item=Synergy-Enc1-2, bay 3)
changed: [localhost] => (item=Synergy-Enc1-1, bay 3)

PLAY [Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template in OneView] *****

TASK [Creating Server Profile [RH-1] from Server Profile Template [HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template]] **
changed: [RH-2]
changed: [RH-1]

TASK [Task result of the Server Profile(s) creation] *****
ok: [RH-1] => {
    "msg": "Server Profile created."
}
ok: [RH-2] => {
    "msg": "Server Profile created."
}

TASK [Powering on the Compute Module(s) [Synergy-Enc1-2, bay 3]] *****
changed: [RH-1] ①
changed: [RH-2] ②

TASK [debug] *****
ok: [RH-1] => {
    "msg": "The server is located in Synergy-Enc1-2, bay 3"
}
ok: [RH-2] => {
    "msg": "The server is located in Synergy-Enc1-3, bay 3" ③
}

PLAY RECAP *****
RH-1           : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
RH-2           : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost      : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

lione1@MIC2FZV4RN:~$ 
```

- ①: Task result of the server profiles creation.
- ②: Task result of the powering-on of the server hardware.
- ③: Displays the server hardware location used by the server profiles

- Once completed, two Server Profiles RH-1 and RH-2 are created, and the servers are powered-on:

The screenshot shows the HPE OneView interface for managing server profiles. On the left, there's a sidebar titled "Server Profiles 5" with a green button labeled "+ Create profile". Below it is a list of profiles: Profile-1, Profile-2, Profile-3, RH-1 (selected and highlighted with a red box), and RH-2. The main panel shows the details for the selected profile, "RH-1". The "General" tab is active. The profile description is "Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows". It uses the "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template". The hardware is a "SY 660 Gen9 1" in an "Synergy-Encl-2\_bay\_3" enclosure group, located in "EG-Synergy-Local". The device bay is "On". The server power is "On". The serial number is "VCG0NC3004", the UUID is "9df6a644-050e-4e0d-b712-7284a05cd784", and the iSCSI initiator name is "iqn.2015-02.com.hpe:oneview-vcg0nc3004".

This concludes Ansible - Scenario 2

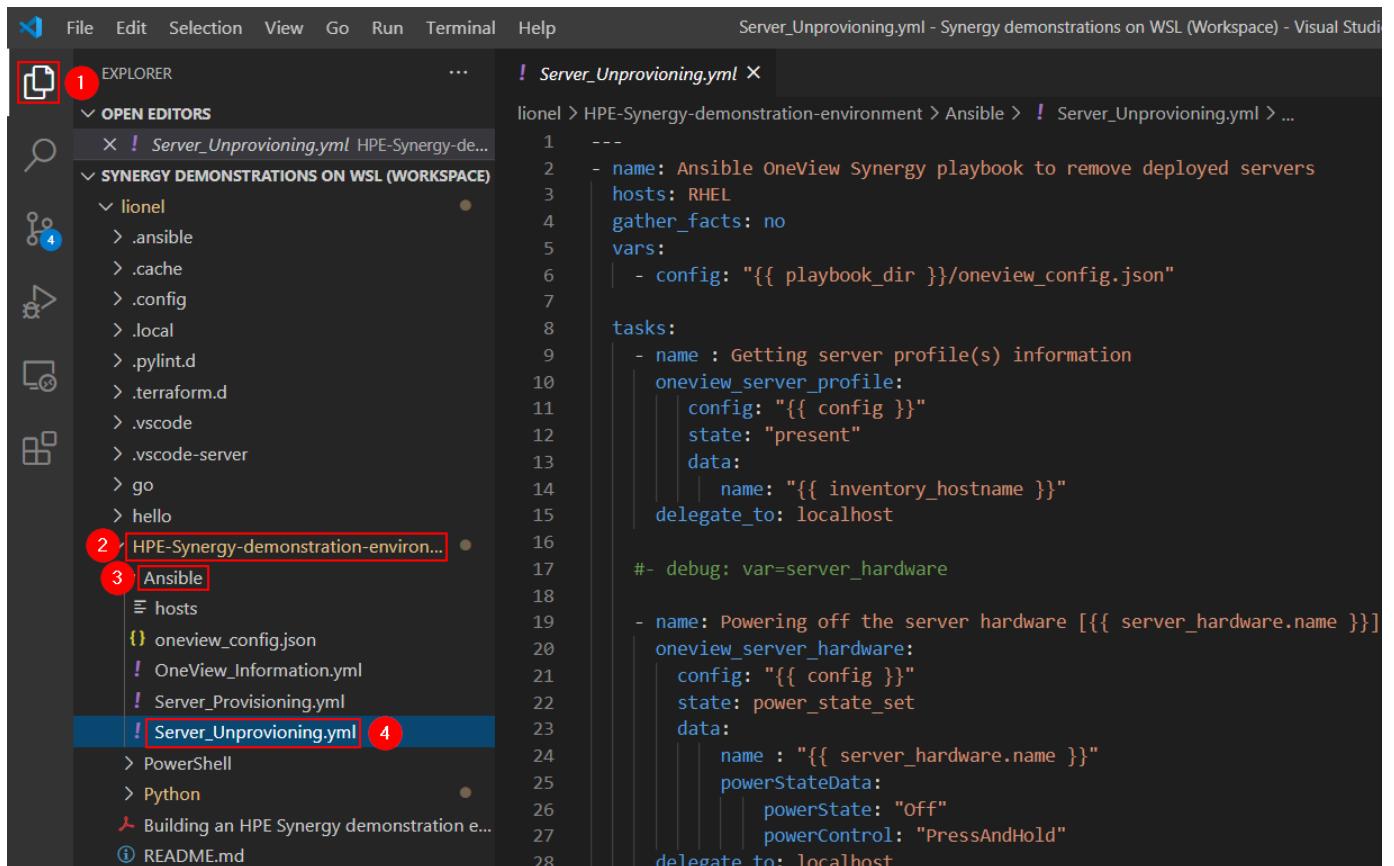


## Ansible – Scenario 3 – Unprovisioning Running Servers

Now that the two servers that have been created in scenario 2 are running, we can explain to the customer that as easily as the provisioning has been made, we can now unprovision the servers and return the two server hardware back to the OneView resource pool.

**Note:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available) AND after Ansible – Scenario 2 when server profiles *RH-1* and *RH-2* are available in OneView.

- Open VS Code using the **Synergy demonstrations on WLS** workspace
- in Explorer, open the Ansible playbook `Server_Unprovisioning.yml` located in the **Ansible** folder of the **HPE-Synergy-demonstration-environment** repository:



The screenshot shows the Visual Studio Code interface. The left sidebar (Explorer) displays a tree structure of files and folders. The main area (Editor) shows the content of the `Server_Unprovisioning.yml` file. The file contains Ansible YAML code for unprovisioning servers. Red circles with numbers 1 through 4 highlight specific items in the Explorer and Editor:

- 1: The `Server_Unprovisioning.yml` file in the Explorer.
- 2: The `SYNTERGY DEMONSTRATIONS ON WSL (WORKSPACE)` workspace in the Explorer.
- 3: The `Ansible` folder in the Explorer.
- 4: The `Server_Unprovisioning.yml` file in the Editor.

```

---  

- name: Ansible OneView Synergy playbook to remove deployed servers  

hosts: RHEL  

gather_facts: no  

vars:  

| - config: "{{ playbook_dir }}/oneview_config.json"  

tasks:  

- name : Getting server profile(s) information  

  oneview_server_profile:  

| config: "{{ config }}"  

| state: "present"  

| data:  

|   name: "{{ inventory_hostname }}"  

delegate_to: localhost  

#- debug: var=server_hwreare  

- name: Powering off the server hardware [{{ server_hardware.name }}]  

  oneview_server_hardware:  

| config: "{{ config }}"  

| state: power_state_set  

| data:  

|   name : "{{ server_hardware.name }}"  

|   powerStateData:  

|     powerState: "off"  

|     powerControl: "PressAndHold"  

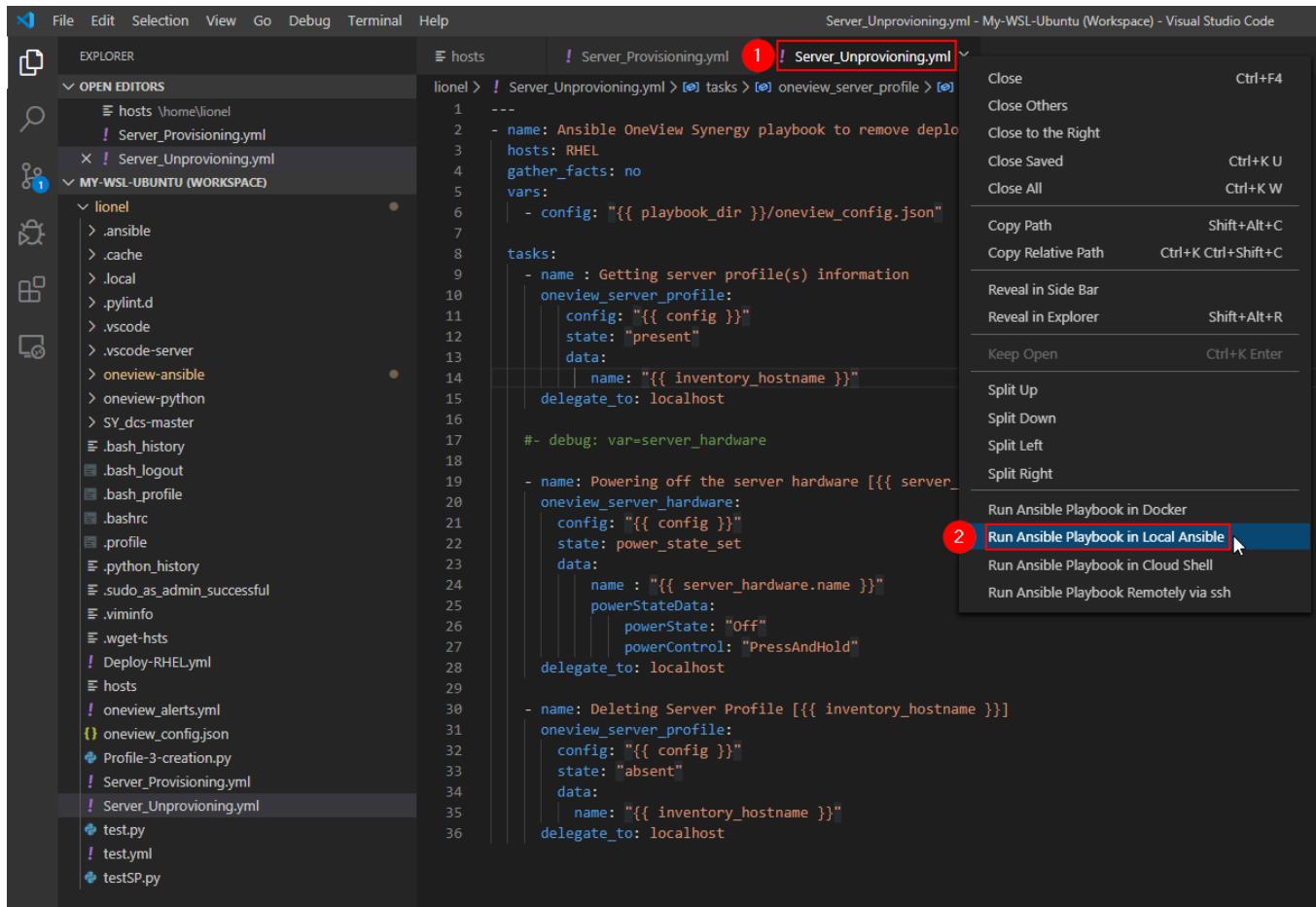
delegate to: localhost

```

This playbook has three tasks:

1. Getting server profiles information using the *hosts* file inventory information.
2. Powering off the server hardware using the name collected from the server profile
3. Deleting the server profiles using the name pulled from the *hosts* files.

- To run the playbook, right click on the file tab then select **Run Ansible Playbook in Local Ansible**



- Both servers should power-off and the server profiles should be erased.

The screenshot shows the HPE OneView interface for managing server profiles. On the left, there is a list of profiles: Profile-1, Profile-2, Profile-3, RH-1, and RH-2. RH-1 is selected and highlighted with a red box. The main panel displays the 'General' tab for RH-1. A progress bar at the top indicates 'Clearing server hardware settings from Synergy-Encl-2, bay 3.' The detailed configuration includes:

Description	Value
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	<a href="#">Synergy-Encl-2, bay 3</a>
Server hardware type	<a href="#">SY 660 Gen9.1</a>
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	Device bay
Server power	Off
Serial number (v)	VCG0NC3004
UUID (v)	9df6a644-050e-4e0d-b712-7284a05cd784
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg0nc3004

- After a few seconds, the server profiles are deleted:

The screenshot shows the HPE OneView interface after the server profiles have been deleted. The left sidebar shows 'Server Profiles 0'. The main panel shows a single entry for RH-1, which has been deleted, indicated by a red circle with a question mark icon. The status message 'Completed 28s' and the administrator information 'administrator 2/25/20 11:13:57 am' are displayed next to it.

- In the terminal, the following outputs get displayed:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

lionel@MIC2FZV4RN:~$ export VS CODEEXT_USER_AGENT=vsco ss.vscod e-ansib le-0.5.2
lionel@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/Server_Unprovisioning.yml"

PLAY [Ansible OneView Synergy playbook to remove deployed servers] ****
TASK [Getting server profile(s) information] ****
ok: [RH-2]
ok: [RH-1]

TASK [Powering off the server hardware [Synergy-Encl-2, bay 3]] ****
changed: [RH-1] ①
changed: [RH-2] ②

TASK [Deleting Server Profile [RH-1]] ****
changed: [RH-1]
changed: [RH-2]

PLAY RECAP ****
RH-1 : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
RH-2 : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

lionel@MIC2FZV4RN:~$
```

- ①: Task result of the server hardware powering-off.
- ②: Task result of the server profiles deletion.

This concludes Ansible – Scenario 3



## Terraform – Scenario 1 – Provisioning a new server

Terraform, by HashiCorp, is a popular tool for infrastructure automation that facilitates a multi-cloud or hybrid cloud approach to infrastructure management. When integrated with HPE OneView, Terraform is a powerful orchestration tool that can be used to create, manage, and update infrastructure resources. These resources may be server profiles, server hardware, logical enclosure, networks, and so on.

Terraform is a tool not only for building and changing but also for versioning infrastructure. Configuration files that describe actions over components are used to generate an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is also able to determine what changed and create incremental execution plans which can be applied.

HPE OneView Terraform provider automates the provisioning of physical infrastructure on-demand using Software-Defined templates from HPE OneView. This enables administrators to create a resource topology similar to that of a public cloud on their own physical infrastructure. With this type of resource topology, administrators can easily migrate applications and workloads to an on-premises private cloud environment to realize their hybrid cloud strategy.

**Note:** To learn more about Terraform with HPE OneView, see <https://github.com/HewlettPackard/terraform-provider-oneview>

In this scenario, we are going to use Terraform to automate the creation of a server profile using an existing Server Profile Template.

**Important notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available)

To automate the creation of a server profile using Terraform, we need first to create a Terraform project (a folder) with a Terraform configuration file:

- From the Ubuntu WSL console and enter:

```
cd ~
```

**Important note:** It is very important to run the following commands from the Ubuntu WSL console and not from the VS Code one as you will face an illegal char error message when performing the *terraform init* command. This is due to a file encoding format produced by VS Code that Terraform is unable to read.

```
lionel@MIC2FZV4RN:~/terraform-create-profile $ terraform init  
There are some problems with the configuration, described below.
```

The Terraform configuration must be valid before initialization so that  
Terraform can determine which modules and providers need to be installed.

Error: Error parsing /home/lionel/terraform-create-profile/create-profile.tf: At 2:9: illegal char

- Then create a new directory named `terraform-create-profile`

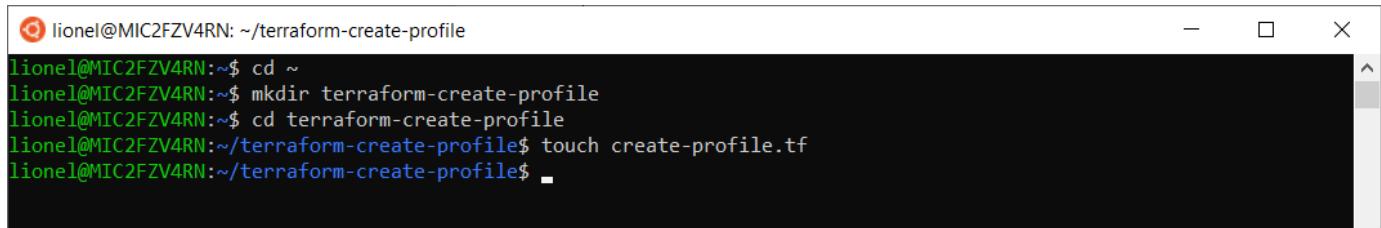
```
mkdir terraform-create-profile
```

- Change to the directory:

```
cd terraform-create-profile
```

- Then create a new empty file using `touch` to create our Terraform configuration file:

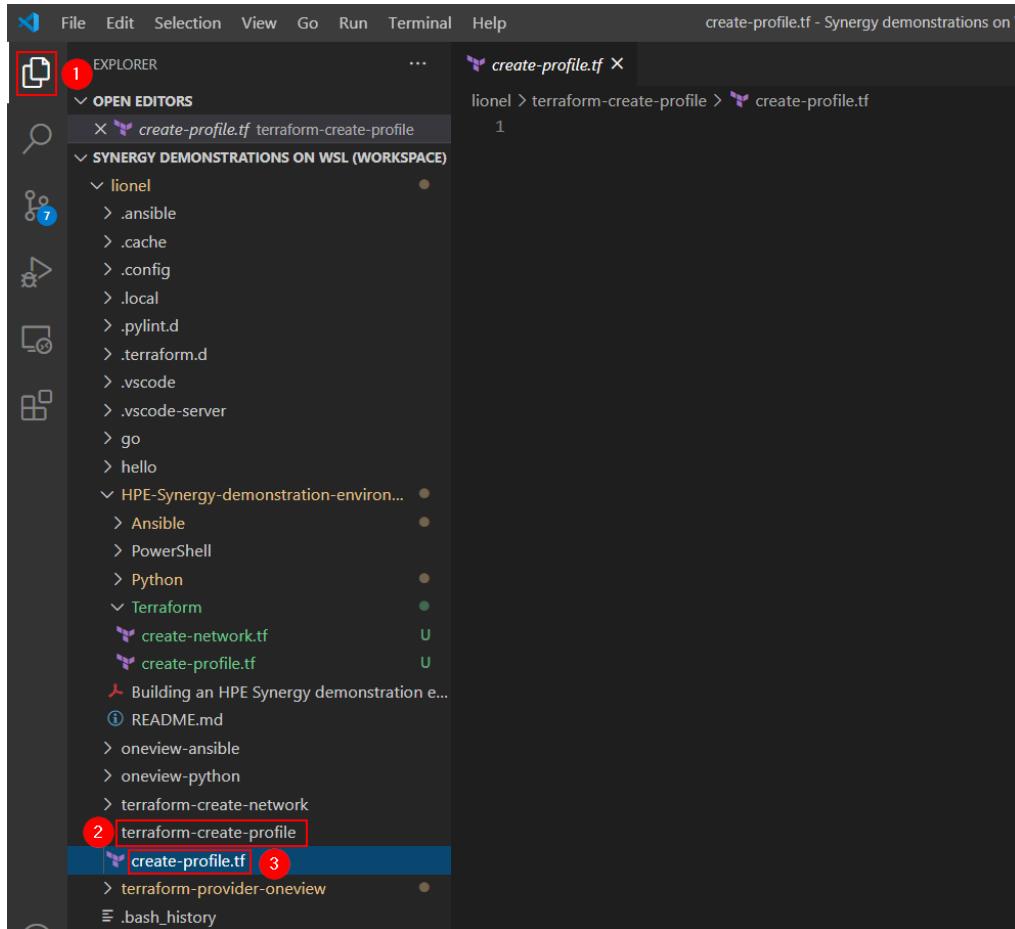
```
touch create-profile.tf
```



A screenshot of a terminal window titled "lionel@MIC2FZV4RN: ~/terraform-create-profile". The window shows the following command history:

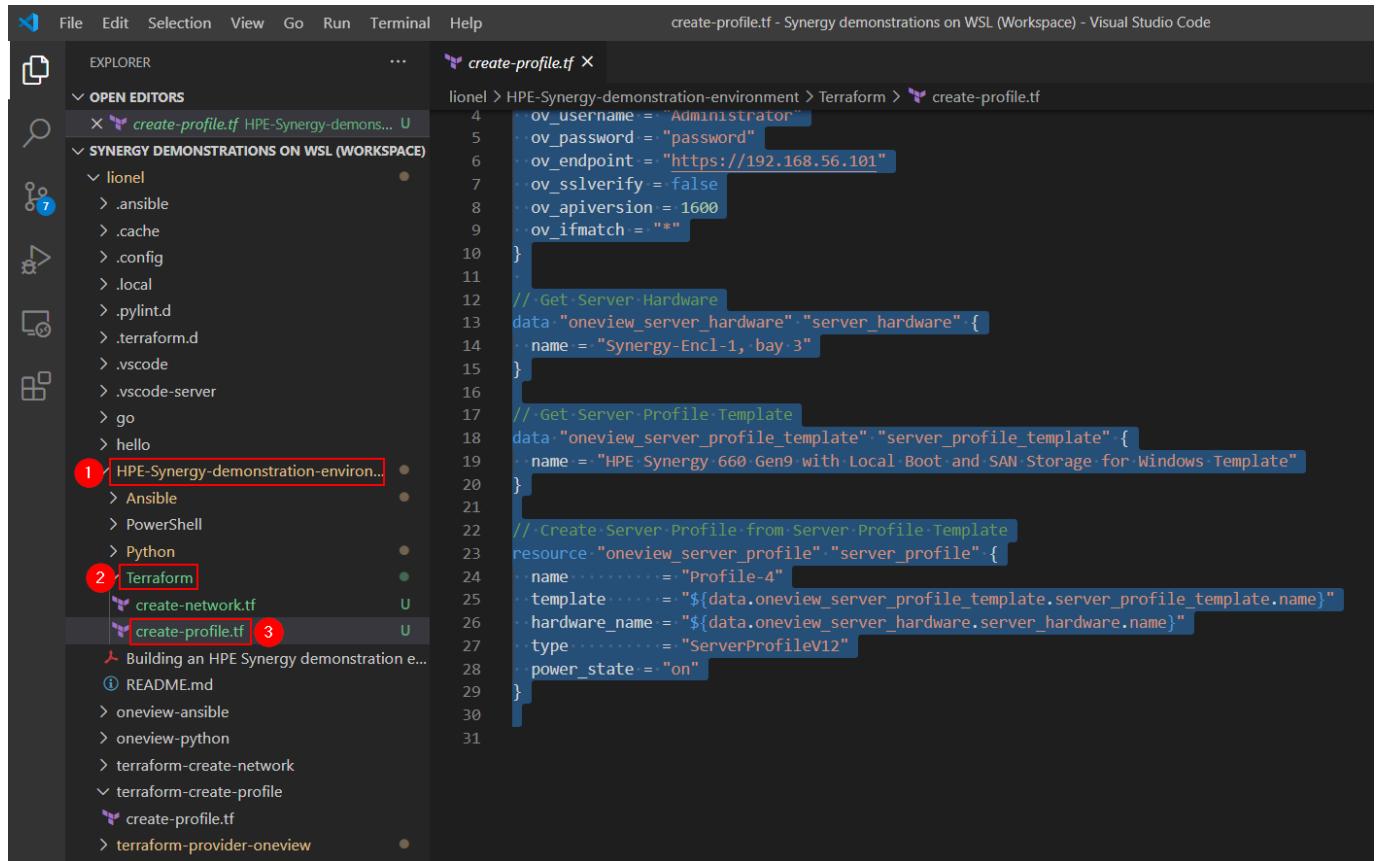
```
lionel@MIC2FZV4RN:~$ cd ~
lionel@MIC2FZV4RN:~$ mkdir terraform-create-profile
lionel@MIC2FZV4RN:~$ cd terraform-create-profile
lionel@MIC2FZV4RN:~/terraform-create-profile$ touch create-profile.tf
lionel@MIC2FZV4RN:~/terraform-create-profile$
```

- After that, you can move back to the VS Code console and open `create-profile.tf` in the **terraform-create-profile** folder:



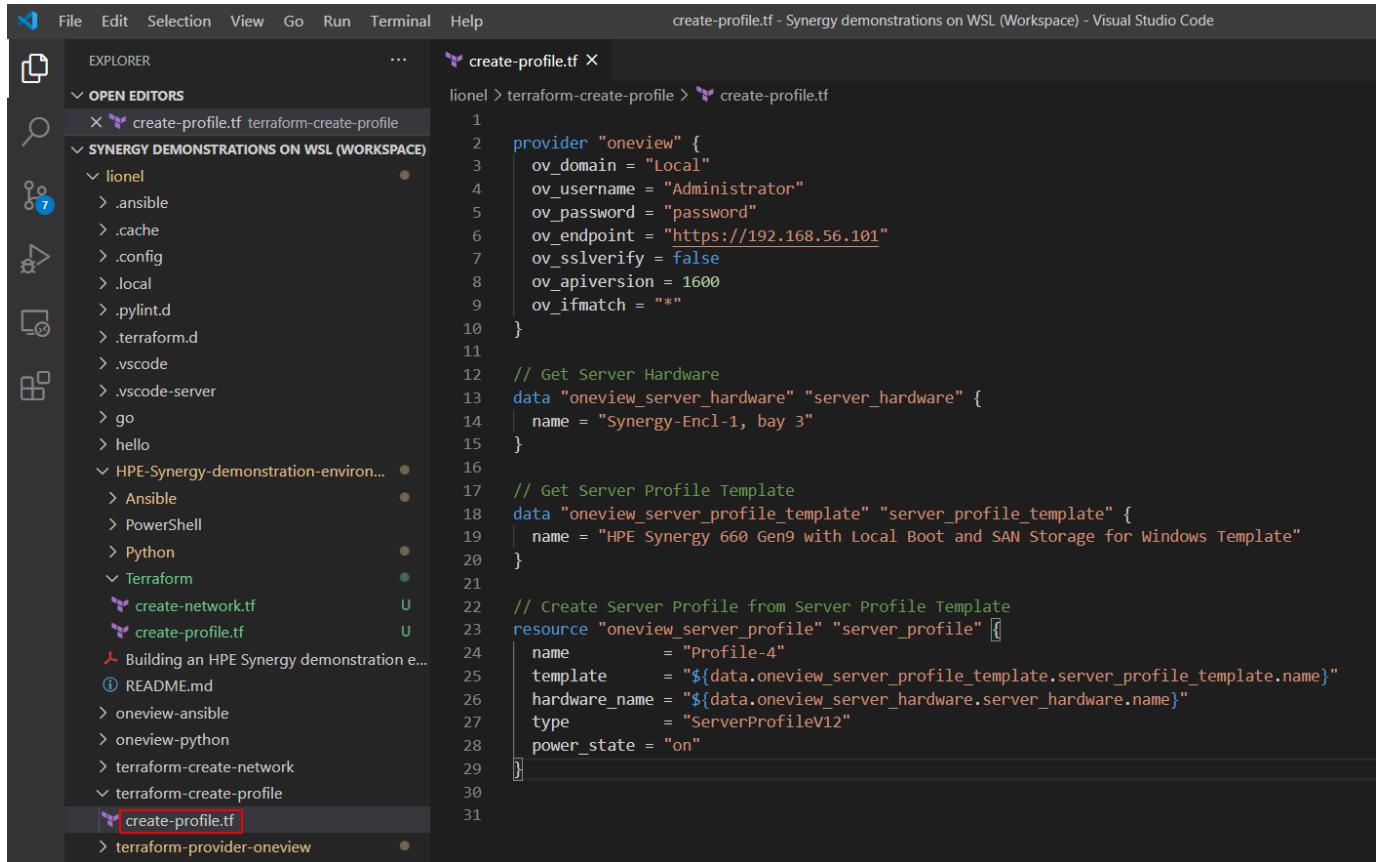
- The file is currently empty, so we need to copy/paste the content of the `create-profile.tf` available in the **HPE-Synergy-demonstration-environment** repository in our new file.

Open `create-profile.tf` located in the **Terraform** folder of the **HPE-Synergy-demonstration-environment** repository then press **CRTL + A** then **CTRL + C**



```
4   ov_username = "Administrator"
5   ov_password = "password"
6   ov_endpoint = "https://192.168.56.101"
7   ov_sslverify = false
8   ov_apiversion = 1600
9   ov_ifmatch = "*"
10 }
11
12 // Get Server Hardware
13 data "oneview_server_hardware" "server_hardware" {
14   name = "Synergy-Encl-1, bay 3"
15 }
16
17 // Get Server Profile Template
18 data "oneview_server_profile_template" "server_profile_template" {
19   name = "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
20 }
21
22 // Create Server Profile from Server Profile Template
23 resource "oneview_server_profile" "server_profile" {
24   name        = "Profile-4"
25   template    = "${data.oneview_server_profile_template.server_profile_template.name}"
26   hardware_name = "${data.oneview_server_hardware.server_hardware.name}"
27   type        = "ServerProfileV12"
28   power_state = "on"
29 }
30
31 }
```

- Then paste all content to our new file using **CTRL + V** then save the file by pressing **CTRL + S**



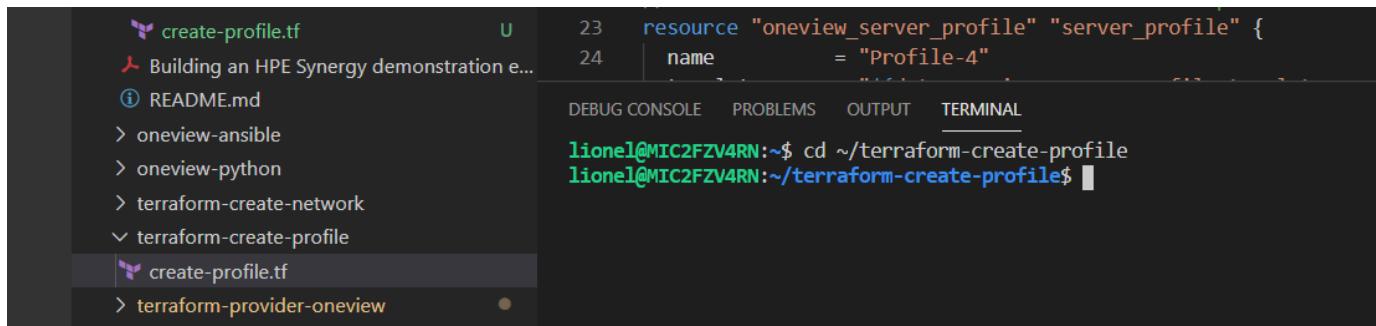
```

1 provider "oneview" {
2   ov_domain = "Local"
3   ov_username = "Administrator"
4   ov_password = "password"
5   ov_endpoint = "https://192.168.56.101"
6   ov_sslverify = false
7   ov_apiversion = 1600
8   ov_ifmatch = "*"
9 }
10
11 // Get Server Hardware
12 data "oneview_server_hardware" "server.hardware" {
13   name = "Synergy-Encl-1, bay 3"
14 }
15
16 // Get Server Profile Template
17 data "oneview_server_profile_template" "server.profile.template" {
18   name = "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
19 }
20
21
22 // Create Server Profile from Server Profile Template
23 resource "oneview_server_profile" "server.profile" {
24   name        = "Profile-4"
25   template    = "${data.oneview_server_profile_template.server.profile.template.name}"
26   hardware_name = "${data.oneview_server_hardware.server.hardware.name}"
27   type        = "ServerProfileV12"
28   power_state = "on"
29 }
30
31

```

- Then from the VS Code console, go to the *terraform-create-profile* folder:

```
cd ~/terraform-create-profile
```



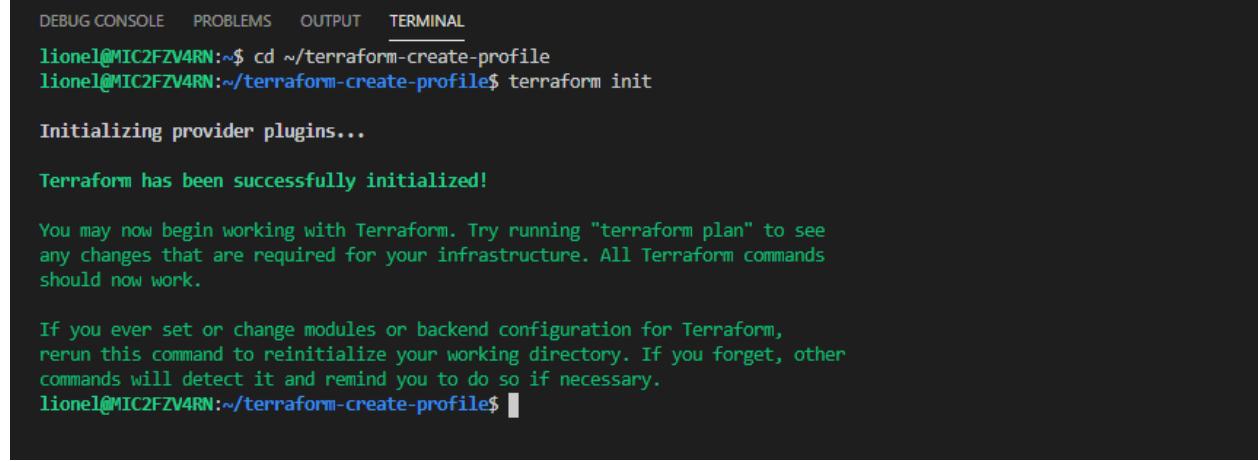
```

lione1@MIC2FZV4RN:~$ cd ~/terraform-create-profile
lione1@MIC2FZV4RN:~/terraform-create-profile$ 

```

- Now we need to run the *init* command which will prepare Terraform in this directory:

```
terraform init
```



The screenshot shows a terminal window with tabs for DEBUG CONSOLE, PROBLEMS, OUTPUT, and TERMINAL. The TERMINAL tab is active. The terminal output is as follows:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
lionel@MIC2FZV4RN:~$ cd ~/terraform-create-profile
lionel@MIC2FZV4RN:~/terraform-create-profile$ terraform init

Initializing provider plugins...
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
lionel@MIC2FZV4RN:~/terraform-create-profile$
```

- Next, we can verify that Terraform will create the server profile resource as we expect before making any actual changes to our infrastructure, enter:

```
terraform plan
```

The screenshot shows a terminal window with tabs for DEBUG CONSOLE, PROBLEMS, OUTPUT, and TERMINAL. The TERMINAL tab is active, displaying the following output:

```
lionel@MIC2FZV4RN:~/terraform-create-profile$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

data.oneview_server_hardware.server_hardware: Refreshing state...
data.oneview_server_profile_template.server_profile_template: Refreshing state...

-----
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

+ oneview_server_profile.server_profile
  id:          <computed>
  affinity:    "Bay"
  hardware_name: "Synergy-Encl-1, bay 3"
  hardware_uri: <computed>
  hide_unused_flex_nics: "true"
  ilo_ip:      <computed>
  mac_type:    "Virtual"
  name:        "Profile-4"
  power_state: "on"
  public_mac:  <computed>
  public_slot_id: <computed>
  serial_number: <computed>
  serial_number_type: "Virtual"
  template:    "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
  type:        "ServerProfileV12"
  uri:         <computed>
  wwn_type:    "Virtual"

Plan: 1 to add, 0 to change, 0 to destroy.

-----
Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

lionel@MIC2FZV4RN:~/terraform-create-profile$
```

As we can see the only pending action is the creation of the profile.

- Now that we have evaluated the plan, we can run an apply:

```
terraform apply
```

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

lionel@MIC2FZV4RN:~/terraform-create-profile$ terraform apply
data.oneview_server_profile_template.server_profile_template: Refreshing state...
data.oneview_server_hardware.server_hardware: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

+ oneview_server_profile.server_profile
  id:          <computed>
  affinity:    "Bay"
  hardware_name: "Synergy-Encl-1, bay 3"
  hardware_uri: <computed>
  hide_unused_flex_nics: "true"
  ilo_ip:      <computed>
  mac_type:    "Virtual"
  name:        "Profile-4"
  power_state: "on"
  public_mac:  <computed>
  public_slot_id: <computed>
  serial_number: <computed>
  serial_number_type: "Virtual"
  template:    "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
  type:        "ServerProfileV12"
  uri:         <computed>
  wwn_type:    "Virtual"

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: [
```

- When prompted, type **yes** to confirm that you want to perform the server profile creation.

```
Enter a value: yes

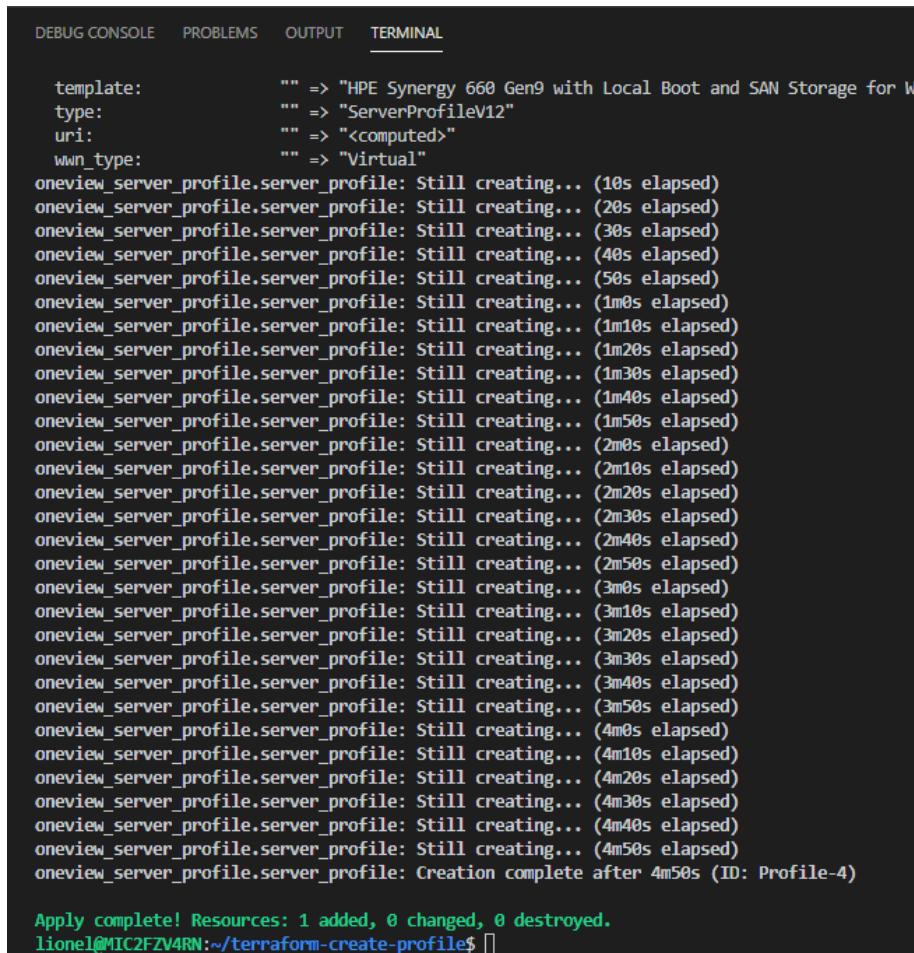
oneview_server_profile.server_profile: Creating...
affinity:      "" => "Bay"
hardware_name: "" => "Synergy-Encl-1, bay 3"
hardware_uri:  "" => "<computed>"
hide_unused_flex_nics: "" => "true"
ilo_ip:        "" => "<computed>"
mac_type:      "" => "Virtual"
name:          "" => "Profile-4"
power_state:   "" => "on"
public_mac:    "" => "<computed>"
public_slot_id: "" => "<computed>"
serial_number: "" => "<computed>"
serial_number_type: "" => "Virtual"
template:      "" => "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
type:          "" => "ServerProfileV12"
uri:           "" => "<computed>"
wwn_type:      "" => "Virtual"

oneview_server_profile.server_profile: Still creating... (10s elapsed)
oneview_server_profile.server_profile: Still creating... (20s elapsed)
oneview_server_profile.server_profile: Still creating... (30s elapsed)
oneview_server_profile.server_profile: Still creating... (40s elapsed)
oneview_server_profile.server_profile: Still creating... (50s elapsed)
```

- Open the OneView UI (<https://192.168.56.101/#/profiles>) to show the creation of the new server profile:

The screenshot shows the OneView interface for managing server profiles. On the left, a sidebar lists existing profiles: Profile-1, Profile-2, Profile-3, Profile-4 (which is selected and highlighted in green), RH-1, and RH-2. A 'Create profile' button is also visible. The main pane displays the details for 'Profile-4'. At the top, there's a progress bar labeled 'Create' with a message: 'Applying server hardware settings to Synergy-Encl-1, bay 3.'. Below this, the 'General' section shows the profile template as 'HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template'. Other sections include 'Connections' (4 OK), 'SAN Storage' (1 Unknown), and 'Local Storage' (1 Unknown). The 'Firmware' section indicates 'managed manually'. At the bottom, an 'ILO Settings' section is shown as 'managed manually'.

- Once created, go back to the VS Code console to show the result of the Terraform configuration:



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

template:      "" => "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Wi
type:         "" => "ServerProfileV12"
uri:          "" => "<computed>"
wwn_type:     "" => "Virtual"
oneview_server_profile.server_profile: Still creating... (10s elapsed)
oneview_server_profile.server_profile: Still creating... (20s elapsed)
oneview_server_profile.server_profile: Still creating... (30s elapsed)
oneview_server_profile.server_profile: Still creating... (40s elapsed)
oneview_server_profile.server_profile: Still creating... (50s elapsed)
oneview_server_profile.server_profile: Still creating... (1m0s elapsed)
oneview_server_profile.server_profile: Still creating... (1m10s elapsed)
oneview_server_profile.server_profile: Still creating... (1m20s elapsed)
oneview_server_profile.server_profile: Still creating... (1m30s elapsed)
oneview_server_profile.server_profile: Still creating... (1m40s elapsed)
oneview_server_profile.server_profile: Still creating... (1m50s elapsed)
oneview_server_profile.server_profile: Still creating... (2m0s elapsed)
oneview_server_profile.server_profile: Still creating... (2m10s elapsed)
oneview_server_profile.server_profile: Still creating... (2m20s elapsed)
oneview_server_profile.server_profile: Still creating... (2m30s elapsed)
oneview_server_profile.server_profile: Still creating... (2m40s elapsed)
oneview_server_profile.server_profile: Still creating... (2m50s elapsed)
oneview_server_profile.server_profile: Still creating... (3m0s elapsed)
oneview_server_profile.server_profile: Still creating... (3m10s elapsed)
oneview_server_profile.server_profile: Still creating... (3m20s elapsed)
oneview_server_profile.server_profile: Still creating... (3m30s elapsed)
oneview_server_profile.server_profile: Still creating... (3m40s elapsed)
oneview_server_profile.server_profile: Still creating... (3m50s elapsed)
oneview_server_profile.server_profile: Still creating... (4m0s elapsed)
oneview_server_profile.server_profile: Still creating... (4m10s elapsed)
oneview_server_profile.server_profile: Still creating... (4m20s elapsed)
oneview_server_profile.server_profile: Still creating... (4m30s elapsed)
oneview_server_profile.server_profile: Still creating... (4m40s elapsed)
oneview_server_profile.server_profile: Still creating... (4m50s elapsed)
oneview_server_profile.server_profile: Creation complete after 4m58s (ID: Profile-4)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
lionel@MIC2FZV4RN:~/terraform-create-profile$ 
```

- Once completed, a new Server Profile *Profile-4* is created, and the server is powered-on as requested in the Terraform configuration file using `power_state = "on"`

The screenshot shows the HPE OneView interface. On the left, there's a sidebar with a 'Create profile' button and a list of profiles: Profile-1, Profile-2, Profile-3, Profile-4 (which is selected and highlighted with a red border), RH-1, and RH-2. The main area shows a summary of 'Profile-4' with a 'Completed' status and a duration of '4m18s'. Below this, the 'General' tab is selected, displaying various configuration details:

Setting	Value
Description	Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	<a href="#">Synergy-Encl-1_bay_3</a>
Server hardware type	<a href="#">SY 660 Gen9.1</a>
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	<a href="#">Device bay</a>
Server power	<a href="#">On</a>
Serial number (v)	VCG0NC300D
UUID (v)	c5e2da8e-3d6f-497e-abf1-cff9b2661ee1
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg0nc300d

Below the general settings, there's a section titled 'OS Deployment'.

This concludes Terraform – Scenario 2

## Terraform – Scenario 2 – Unprovisioning Running Server

Now that the server that has been created in scenario 1 is up and running, as easily as the provisioning has been made, we can now unprovision the server and return the server hardware back to the OneView resource pool.

**Note:** This scenario can only be run with the Final configuration snapshot (i.e. when a Configured Logical Enclosure is available) AND only after *Terraform – Scenario 2* when server profile *Profile-4* is available in OneView.

To do this using Terraform is incredibly easy as there is no need to create a new configuration file but just to destroy this configuration project using the `destroy` command.

- From the VS Code console, enter:

```
terraform destroy
```

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL 1: terraform + - x
lione1@MIC2FZV4RN:~/terraform-create-profile$ terraform destroy
data.oneview_server_profile_template.server_profile_template: Refreshing state...
data.oneview_server_hardware.server_hardware: Refreshing state...
oneview_server_profile.server_profile: Refreshing state... (ID: Profile-4)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

- oneview_server_profile.server_profile

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: [REDACTED]
```

This command generates a report detailing what actions Terraform will take to destroy our oneview resources. We can see that Terraform will perform 1 action to destroy the server profile resource.

- Enter **Yes** to proceed the destroy action.

- In the first place, the server is powered off:

The screenshot shows the HPE OneView interface for managing server profiles. On the left, a sidebar lists 'Server Profiles' with 6 items: Profile-1, Profile-2, Profile-3, Profile-4 (selected), RH-1, and RH-2. The main panel displays 'Profile-4' in edit mode. The status bar at the top right indicates 'Completed 4m18s'. The 'General' tab is selected, showing the following configuration details:

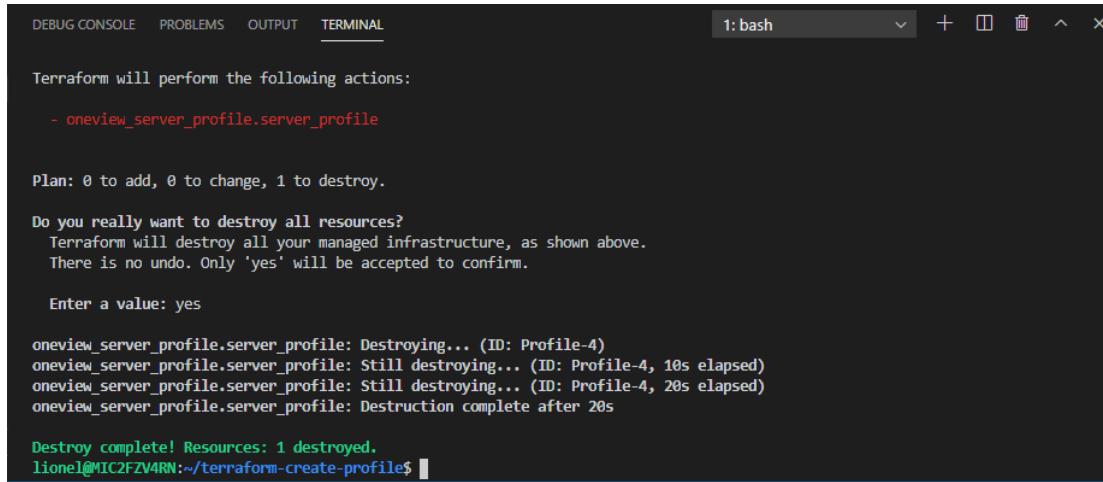
Description	Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	<a href="#">Synergy-Encl-1.bay.3</a>
Server hardware type	<a href="#">SY 660 Gen9.1</a>
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	Device bay
Server power	Off
Serial number (v)	VCGONC300D
UUID (v)	c5e2da8e-3d6f-497e-abf1-cff9b2661ee1
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg0nc300d

- Then the profile is deleted:

The screenshot shows the HPE OneView interface for managing server profiles. The 'Profile-4' entry in the sidebar has a delete icon next to it. The main panel shows 'Profile-4' in delete mode. The status bar at the top right indicates 'Clearing SAN configuration.' The 'General' tab is selected, showing the following configuration details, identical to the previous state:

Description	Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	<a href="#">Synergy-Encl-1.bay.3</a>
Server hardware type	<a href="#">SY 660 Gen9.1</a>
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	Device bay
Server power	Off
Serial number (v)	VCGONC300D
UUID (v)	c5e2da8e-3d6f-497e-abf1-cff9b2661ee1
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg0nc300d

- In the terminal, the destroy is complete message is displayed with 1 resource destroyed:



The screenshot shows a terminal window titled "1: bash". The terminal displays the following output from Terraform:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
1: bash + □ □ ^ X

Terraform will perform the following actions:
- oneview_server_profile.server_profile

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

oneview_server_profile.server_profile: Destroying... (ID: Profile-4)
oneview_server_profile.server_profile: Still destroying... (ID: Profile-4, 10s elapsed)
oneview_server_profile.server_profile: Still destroying... (ID: Profile-4, 20s elapsed)
oneview_server_profile.server_profile: Destruction complete after 20s

Destroy complete! Resources: 1 destroyed.
lione1@MIC2FZV4RN:~/terraform-create-profile$
```

This concludes Terraform – Scenario 2



We have reached the end of our scenarios to demonstrate a Synergy Infrastructure programmability, infrastructure as-code and all the good benefits and features of our Unified API.

In the next chapter, we are going to reset our environment to prepare our next customer visit.



## Chapter-8 - How to reset the Synergy demonstration environment

We have successfully completed all our live demos and we need now to reset our environment to be ready for our next customer live demonstrations.

The procedure is simple as we have prepared two snapshots to return easily and quickly to a previous state of our DCS appliance. As a reminder, each snapshot provides a different use case:

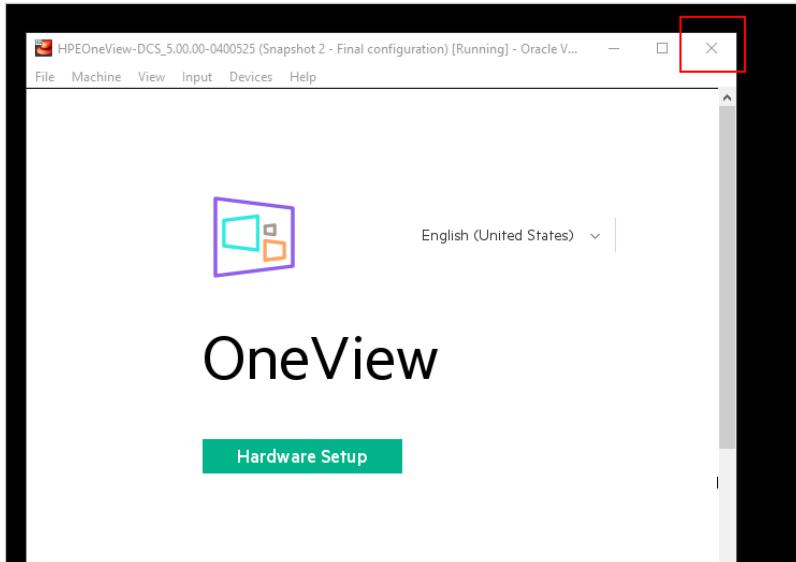
- 1- First snapshot: OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network)
  - ⇒ Can be used to show how to automate the setup of Synergy and the power of our infrastructure as code implementation.
- 2- Second snapshot: OneView appliance is fully configured with LE, EG, LIG and some networks.
  - ⇒ Can be used to run demos with an already configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

### Shutting down the Synergy Composer Demonstration appliance

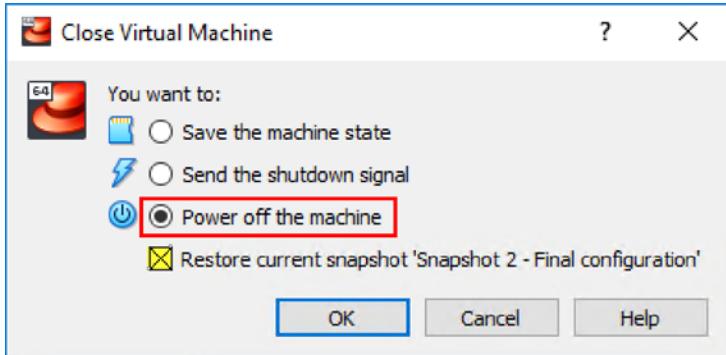
With VirtualBox, when shutting down a virtual machine, you have a "Restore current snapshot" option if you want VirtualBox to load the most recent snapshot the next time you start up the virtual machine again. This is a very convenient and easy option to restore the appliance back to the pre-configured state, just before our live demonstration scenarios.

To restore the appliance back to a pre-demo state:

- Click on the **Close** button in the upper-right corner of the window of the DCS appliance virtual machine



- Select **Power off the machine** and check the restore current snapshot option to restore the **Final configuration** snapshot the next time you start the appliance:



- Click the **OK** button to begin the shutdown process.

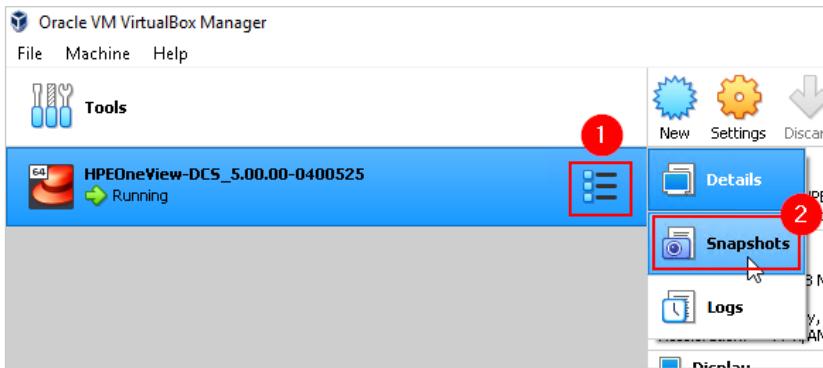
**Note:** By default, VirtualBox shutdown dialog provides a restore snapshot option with the latest created snapshot.

Now the next time you start the DCS appliance, you will be ready to run your live demonstration scenarios again.

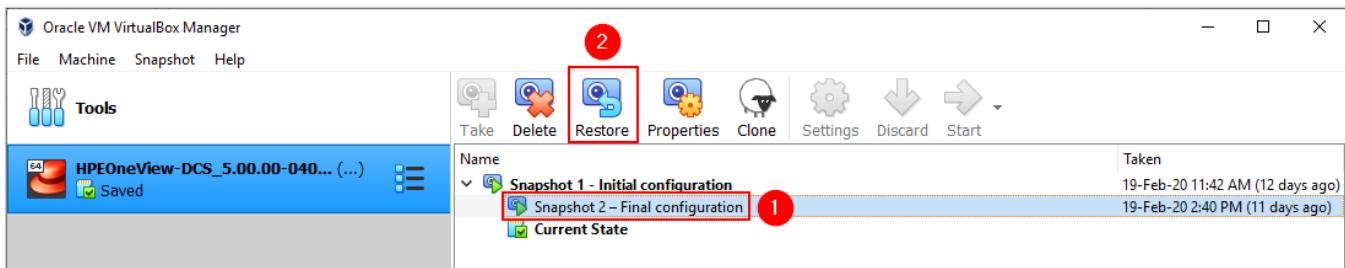
## Resetting the Synergy Composer Demonstration appliance to the fully configured state

To start the appliance with the Synergy frames fully configured (with LE, LIG, EG and networks) in order to run the demos and demonstrate features of the Composable infrastructure, we need to restore the second snapshot.

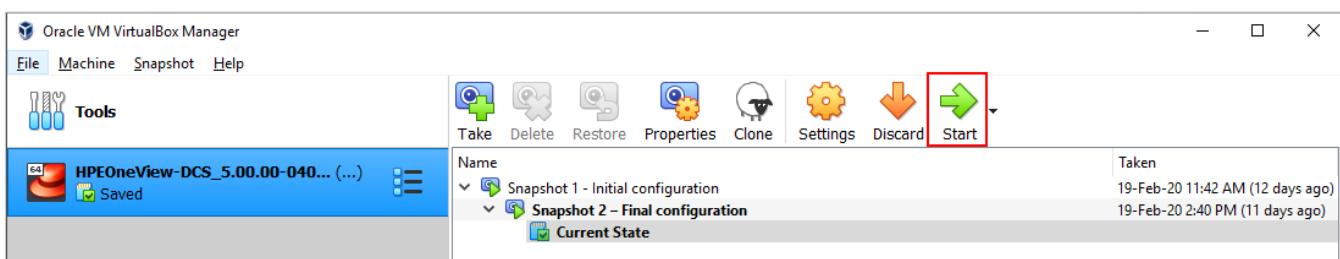
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- Select **Snapshot 2 – Final configuration** then click **Restore**



- Then start the appliance by selecting **Start**



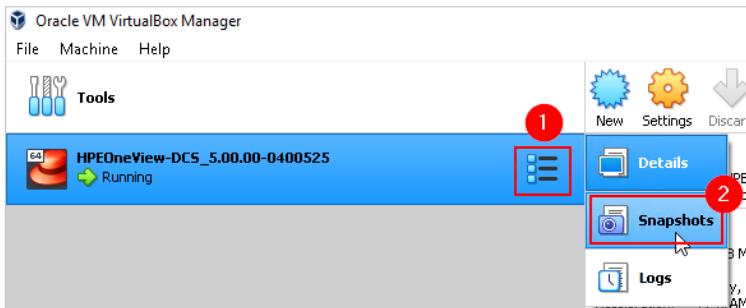
**Note:** With VirtualBox, rolling back to one of the previous snapshots can only be done when a VM is not running.

**Note:** When you start the DCS appliance using the snapshots prepared in this document, you should never see OneView starting and taking several minutes to start. If you see OneView starting, shutdown the VM and start again using the earlier procedure.

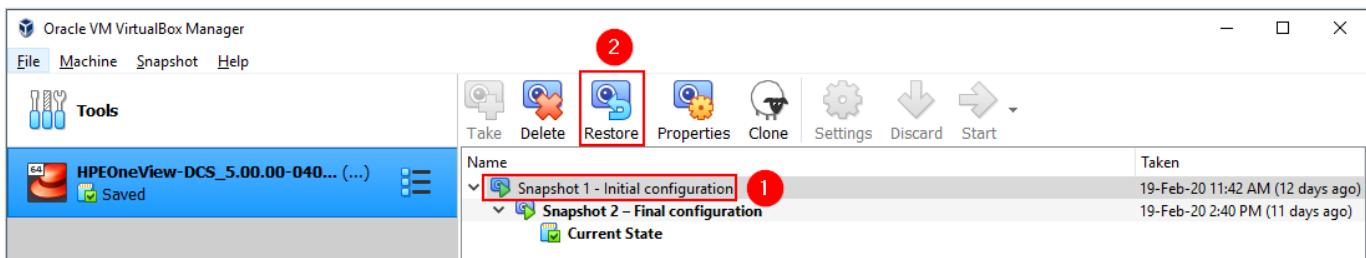
## Resetting the Synergy Composer Demonstration appliance to the unconfigured state

To start the appliance with the Synergy frames unconfigured (no LE, no LIG, no EG, no network) in order to show how to automate the setup of Synergy, we need to restore the first snapshot.

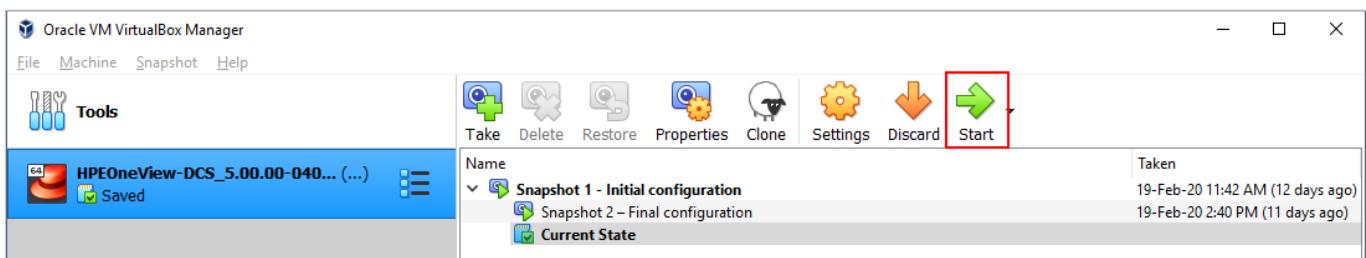
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- Select **Snapshot 1 – Initial configuration** then click **Restore**



- Then start the appliance by selecting **Start**



**Note:** With VirtualBox, rolling back to one of the previous snapshots can only be done when a VM is not running.

**Note:** When you start the DCS appliance using the snapshots prepared in this document, you should never see OneView starting and taking several minutes to start. If you see OneView starting, shutdown the VM and start again using the earlier procedure.

## Chapter-9 - How to upgrade the Synergy demonstration environment

This chapter describes how you can upgrade the different components of your Synergy demonstration environment.

### Upgrading the HPE OneView demonstration appliance (DCS)

It is not supported to upgrade the DCS appliance using the standard OneView upgrade procedure, you need to download a new version of the HPE OneView demonstration appliance and re-install it from scratch using the same steps provided in this guide.

### Upgrading HPE OneView Python SDK

If an old version of the module is already installed on your system, you can use the following commands to update to the latest version:

```
cd $home
sudo rm -r -f oneview-python
git clone https://github.com/HewlettPackard/oneview-python.git
cd oneview-python
python3 setup.py install --user
```

### Upgrading Ansible modules for HPE OneView

If an old version of the module is already installed on your system, you can use the following commands to update to the latest version:

```
cd $home
rm -r -f oneview-ansible
git clone https://github.com/HewlettPackard/oneview-ansible.git
cd oneview-ansible
pip3 install -r requirements.txt
```

**Note:** make sure you also merge your branch with the *bug\_fix/create\_profiles\_in\_parallel* after the upgrade. See [Installing Ansible Modules for HPE OneView on Ubuntu WSL](#)

### Upgrading the Terraform provider for HPE OneView

If an old version of the Terraform provider for HPE OneView is already installed on your system, you can use the following commands to update to the latest version:

```
cd /usr/local/bin
go get -u github.com/HewlettPackard/terraform-provider-oneview
sudo mv $GOPATH/bin/terraform-provider-oneview .
```

To update the Terraform provider for HPE OneView GitHub repository clone from your home directory, enter:

```
cd ~
rm -rdf terraform-provider-oneview
sudo git clone https://github.com/HewlettPackard/terraform-provider-oneview.git
```

## Upgrading the PowerShell library for OneView

If a previous major version of the module is already installed on your system, you can use the following commands to uninstall the old version and install the latest version:

```
Get-Module HPOneView.500 -ListAvailable | Uninstall-Module  
Install-Module -Name HPOneView.520
```

If you are not running the latest major version of the HPOneView module, simply enter:

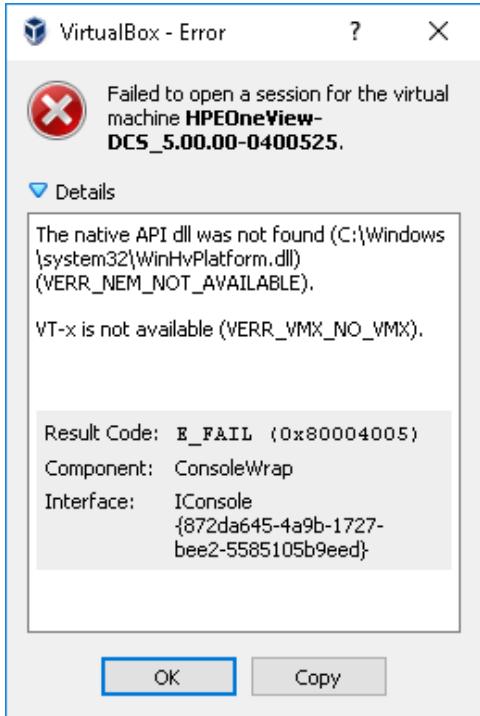
```
Update-Module HPOneView.520 -Force
```



## Troubleshooting VirtualBox VM not starting

### Problem:

You are seeing a **VT-x is not available (VERR\_VMX\_NO\_VMX)** in VirtualBox when starting the VM



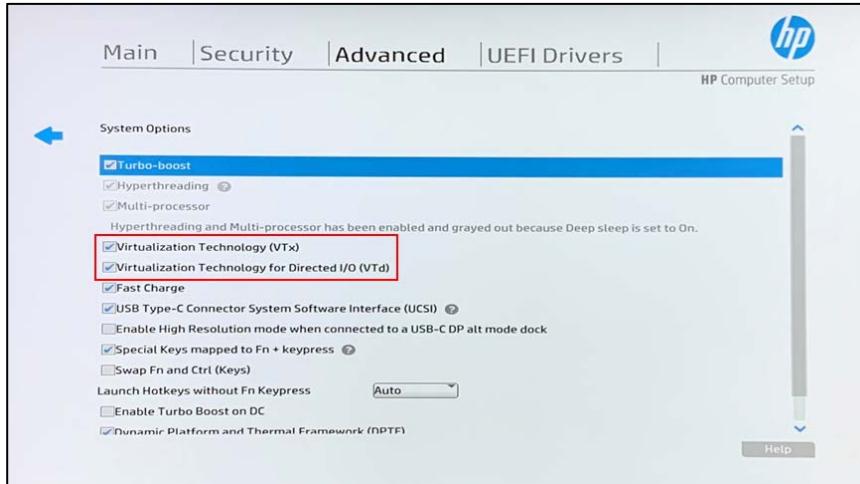
Intel VT-x is a set of processor enhancements to improve virtualization performance.

Oracle VM VirtualBox uses hardware virtualization and requires Intel VT-X to be enabled or not exclusively used by other software.



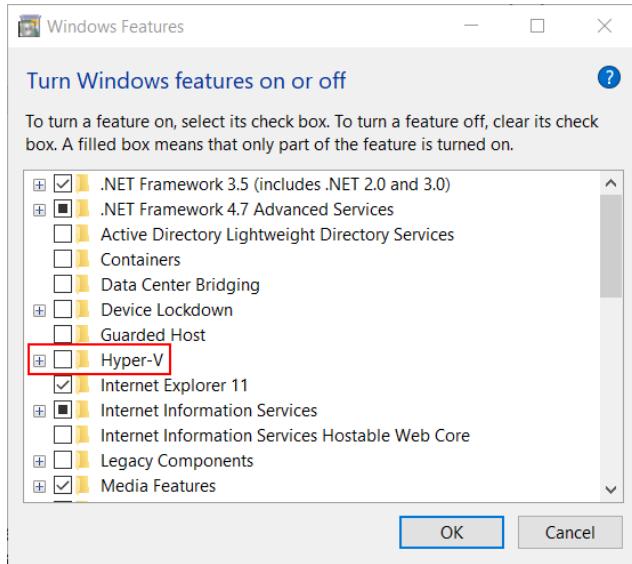
**Solution:**

- Make sure Virtualization (VT-x) is enabled in your BIOS.



**Note:** On HP EliteBook 840 you must repeatedly press the **Esc** key until the Startup Menu opens. Press **F10** to enter the BIOS Setup Utility. Go to **Advanced / System Options**

- Make sure Hyper-V is disabled from Windows features. (Run | OptionalFeatures.exe)

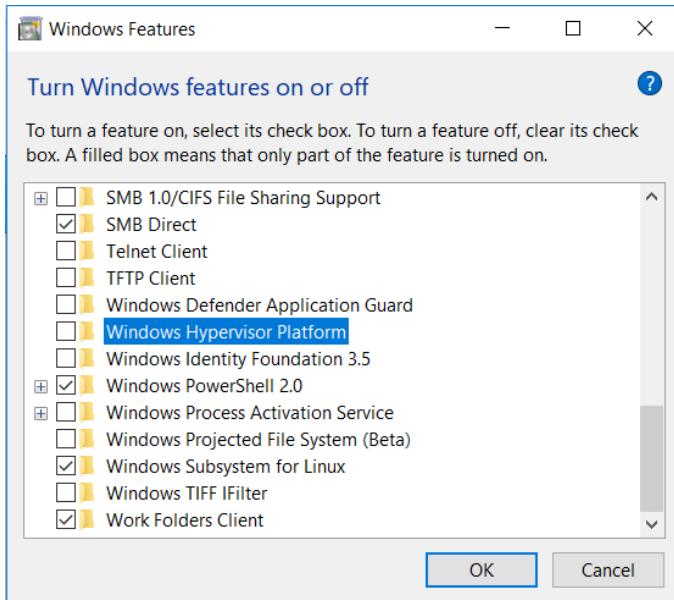


You can also run the following command to disable Hyper-V:

```
dism.exe /Online /Disable-Feature:Microsoft-Hyper-V
```

**Note:** If running, Hyper-V is taking exclusive use of VT-x so it is required to turn off Hyper-V to release VT-x

- Make sure *Windows Hypervisor Platform* is disabled from Windows features (if present).

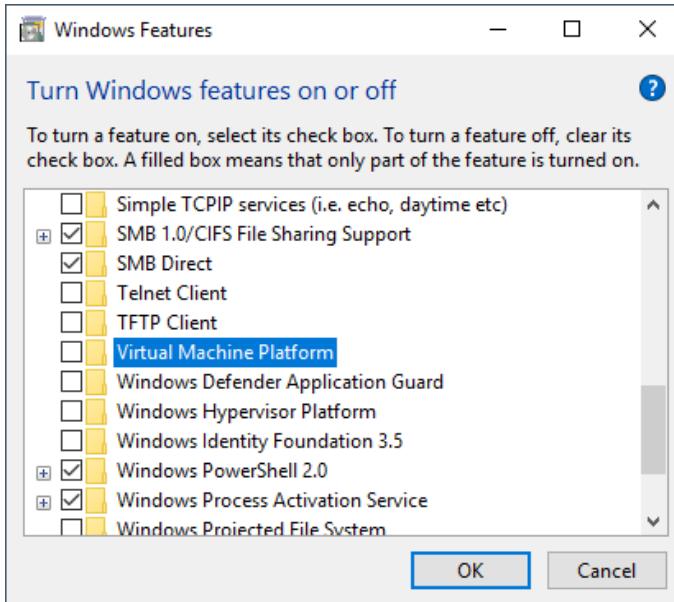


Or run:

```
dism.exe /Online /Disable-Feature:HypervisorPlatform
```

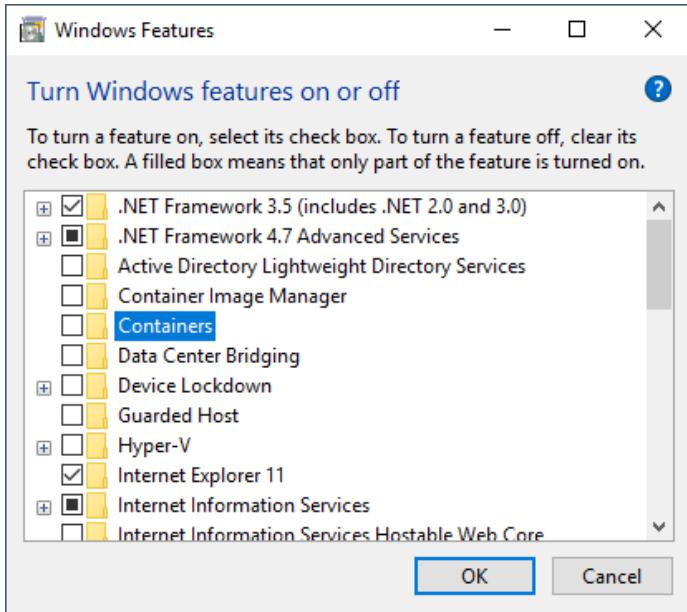
**Note:** If running, *Windows Hypervisor Platform* is taking exclusive use of VT-x so it is required to turn it off

- Make sure *Virtual Machine Platform* is disabled from Windows features (if present).



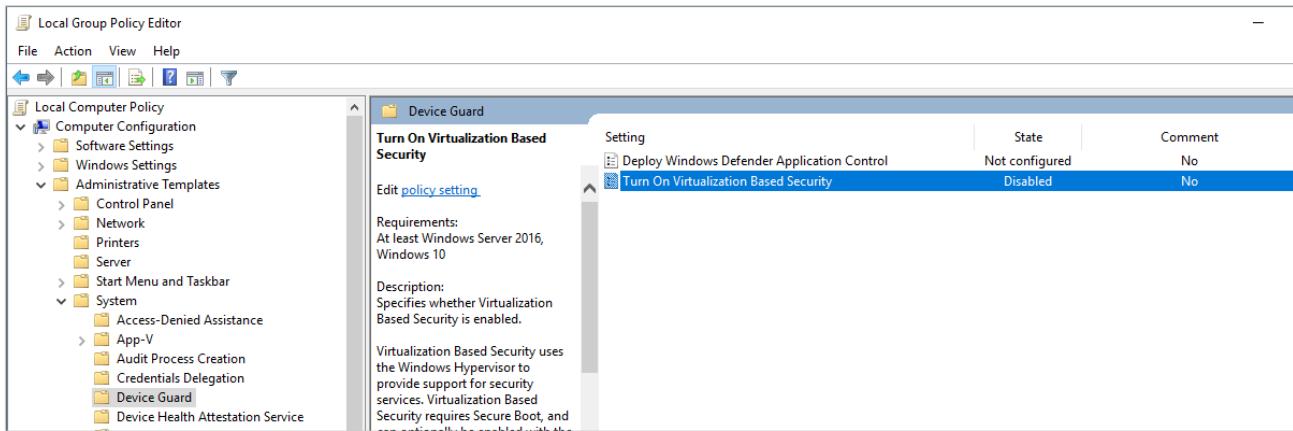
**Note:** If running, *Virtual Machine Platform* is taking exclusive use of VT-x so it is required to turn it off

- Make sure *Container* is disabled from Windows features (if present).



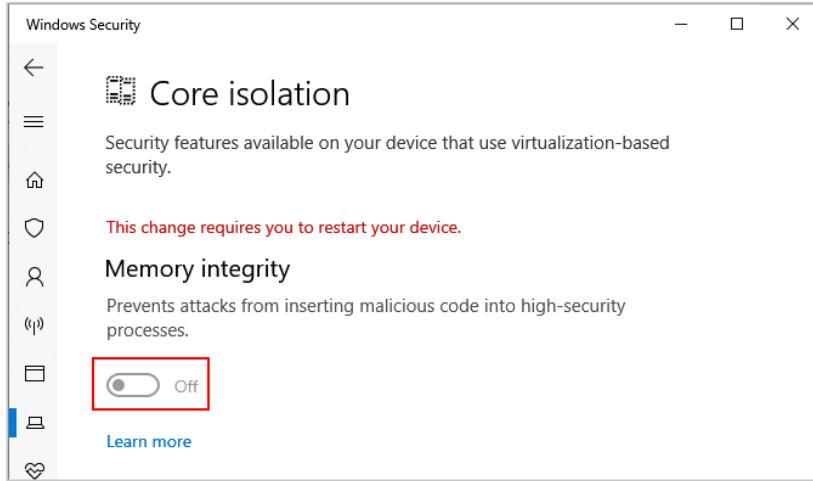
**Note:** If running, *Container* is taking exclusive use of VT-x so it is required to turn it off

- On some Windows hosts with an EFI BIOS, Device Guard or Credential Guard may be active by default and interferes with OS level virtualization apps in the same way that Hyper-V does. These features need to be disabled. On Pro versions of Windows, you can do this using **gpedit.msc** then set **Local Computer Policy > Computer Configuration > Administrative Templates > System > Device Guard > Turn Virtualization Based Security to Disabled**.



If you cannot use gpedit for some reason, then the equivalent registry hack is to find the key `HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\EnableVirtualizationBasedSecurity\Enabled` and set it to 0.

- On Win10 hosts, check **Windows Defender > Device Security > Core Isolation Details** and make sure settings in this panel are turned **off**. A reboot is required to make this change.  
"Core isolation [includes] security features available on your device that use virtualization-based security"



## **Hardware components used to build this demonstration environment**

- HP EliteBook 840 G5
- 32G of RAM
- Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
- System bios Q78 Ver. 01.09.01 - 10/16/2019
- SAMSUNG MZVLW512HMJP 512GB PCI Express 3.0 x4 (NVMe) SSD Drive.



## Summary

The HPE Synergy demonstration environment installation is complete.

Today's Idea Economy is driving IT leaders to find new and innovative ways to deliver the flexibility of hybrid IT solutions while reducing complexity and operating costs. Composable Infrastructure provides the promise of allowing IT to provision and manage traditional workloads along with new mobile and cloud-native applications from a single infrastructure, allowing you to achieve the vision of infrastructure as a code.

You have now the environment to seamlessly prove this and demonstrate the power of a Synergy Composable Infrastructure!

And remember, a product demonstration is one of your best tools to strongly impact your customers to adopt new technologies!

Happy demonstrations!

To help us improve this document, please provide feedback at [lio@hpe.com](mailto:lio@hpe.com).

