



**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
June 2020	<ul style="list-style-type: none"> - HPE OneView DCS 5.20 Z7550-96879 - Windows 10 version 1909 (OS Build 18363.418) - Visual Studio Code version 1.45.1 - PowerShell 7.0.2 - Oracle VM VirtualBox version 6.1.8 r137981 (Qt5.6.2) - Windows Subsystem for Linux with Ubuntu 18.04 LTS - Ansible Modules for HPE OneView 5.6.0 with bug_fix/create_profiles_in_parallel - HPE OneView Python SDK 5.20 - Python 2.7.17 - 3.6.9 - Ansible 2.9.9 - PowerShell for OneView library 5.2.4343.1695 	<ul style="list-style-type: none"> - Update content for HPE Synergy OneView Demonstration appliance - DCS 5.20 - Added Terraform installation in WSL and in VS Code - Added Terraform provider for OneView and Go installation in WSL - Added 3 x Terraform scenarios - Added Chapter-9 - How to upgrade the Synergy demonstration environment - Script improvements for Ansible and Python - Added WSL and Linux distribution recommendations to prevent incompatibility issues - Other minor changes
March 2020	<ul style="list-style-type: none"> - HPE OneView DCS 5.00 Z7550-96681 - Windows 10 version 1809 (OS Build 17763.1039) - Visual Studio Code version 1.42.1 - PowerShell 5.0 - Oracle VM VirtualBox version 6.1.4 r136177 (Qt5.6.2) - Windows Subsystem for Linux with Ubuntu 18.04 LTS - Ansible Modules for HPE OneView 5.4.0 with create_profiles_in_parallel 11/27/19 fix - HPE OneView Python SDK 5.00 - Python 2.7.17 - 3.6.9 - Ansible 2.9.4 - PowerShell for OneView library 5.0.2341.1920 	



ABSTRACT	7
REQUIREMENTS.....	7
DEMONSTRATION APPLIANCE PREREQUISITES.....	8
CHAPTER-1 - DEPLOYING THE HPE ONEVIEW DEMONSTRATION APPLIANCE ON VIRTUALBOX.....	9
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
CHAPTER-2 - PREPARING UBUNTU ON WINDOWS SUBSYSTEM FOR LINUX (WSL)	29
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
CHAPTER-3 - PREPARING MICROSOFT VISUAL STUDIO CODE.....	54
INSTALLING VISUAL STUDIO CODE.....	54
PREPARING VS CODE FOR PYTHON	61
PREPARING VS CODE FOR ANSIBLE	65
PREPARING VS CODE FOR TERRAFORM	66
PREPARING VS CODE FOR POWERSHELL	67
INSTALLING THE POWERSHELL LIBRARY FOR HPE OneVIEW	72
CHAPTER-4 -INITIAL CONFIGURATION OF THE DEMONSTRATION APPLIANCE	74
HARDWARE DISCOVERY AND INITIAL APPLIANCE CONFIGURATION.....	74
CREATION OF AN INITIAL SETUP SNAPSHOT.....	81
CHAPTER-5 - PREPARING POSTMAN	84
INSTALLING AND CONFIGURING POSTMAN.....	84
IMPORTING COLLECTIONS.....	85
CONFIGURATION OF THE POSTMAN ENVIRONMENT	88
CHAPTER-6 - PREPARING YOUR DEMONSTRATION APPLIANCE FOR DEMOS	92
FINAL APPLIANCE CONFIGURATION	93
CREATION OF A FINAL SETUP SNAPSHOT	104
CHAPTER-7 - PREPARING THE LIVE DEMONSTRATION SCENARIOS	107
DEMONSTRATING SYNERGY COMPOSER AND ONEVIEW KEY FEATURES.....	107
DEMONSTRATING INFRASTRUCTURE PROGRAMMABILITY.....	107
Postman - Scenario 1 – Introduction to the OneView REST API.....	108
PowerShell – Scenario 1 - Day-to-day operation task automation	117
PowerShell – Scenario 2 - Creating a report	126
PowerShell – Scenario 3 - Accelerating a configuration change.....	131
Python – Scenario 1 - Day-to-day operation task automation	140
Python – Scenario 2 – Creating a report	146
Python – Scenario 3 - Accelerating a configuration change.....	150
Ansible – Scenario 1 – Collecting facts in OneView	155
Ansible – Scenario 2 – Provisioning New Servers	165
Ansible – Scenario 3 – Unprovisioning Running Servers	177



Terraform – Scenario 1 – Execution plan to accelerate a configuration change.....	181
Terraform – Scenario 2 – Provisioning New Servers.....	198
Terraform – Scenario 3 – Unprovisioning Running Server.....	207
CHAPTER-8 - HOW TO RESET THE SYNERGY DEMONSTRATION ENVIRONMENT	211
SHUTTING DOWN THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE.....	211
RESETTING THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE TO THE FULLY CONFIGURED STATE.....	213
RESETTING THE SYNERGY COMPOSER DEMONSTRATION APPLIANCE TO THE UNCONFIGURED STATE.....	214
CHAPTER-9 - HOW TO UPGRADE THE SYNERGY DEMONSTRATION ENVIRONMENT	215
UPGRADING THE HPE ONEVIEW DEMONSTRATION APPLIANCE (DCS)	215
UPGRADING ANSIBLE MODULES FOR HPE ONEVIEW.....	215
UPGRADING HPE ONEVIEW PYTHON SDK	215
UPGRADING THE POWERSHELL LIBRARY FOR ONEVIEW	216
TROUBLESHOOTING VIRTUALBOX VM NOT STARTING	217
HARDWARE COMPONENTS USED TO BUILD THIS DEMONSTRATION ENVIRONMENT	222
SUMMARY	223



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

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 5.20 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.

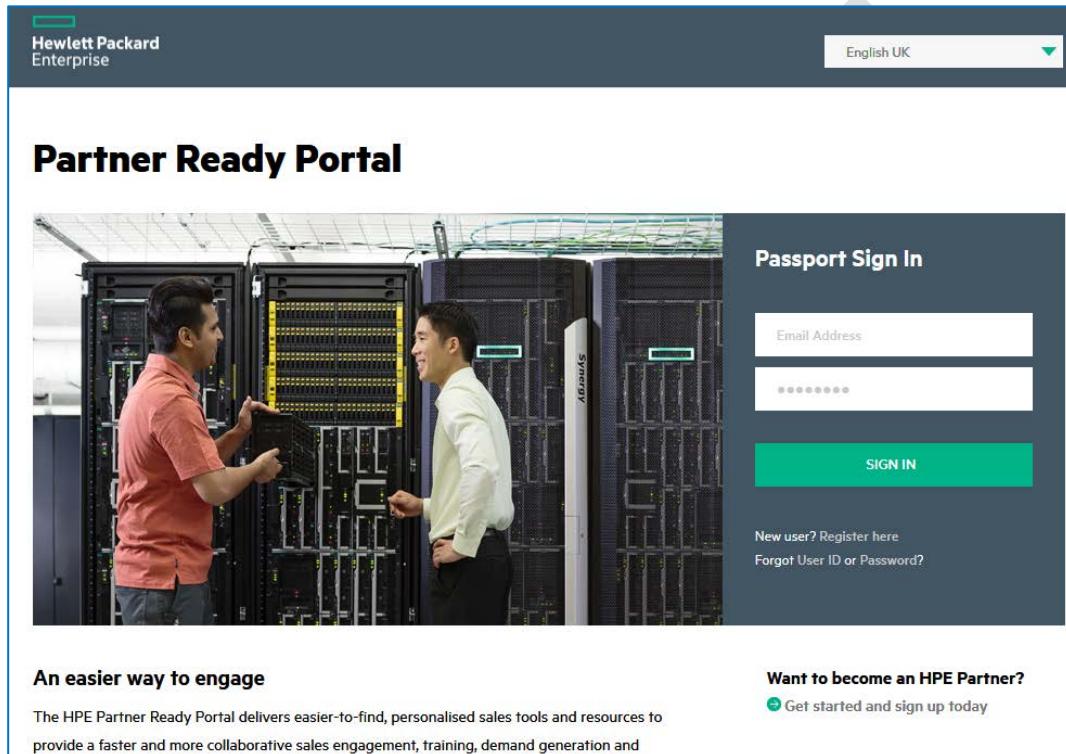


Chapter-1 - Deploying the HPE OneView Demonstration appliance on VirtualBox

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/>

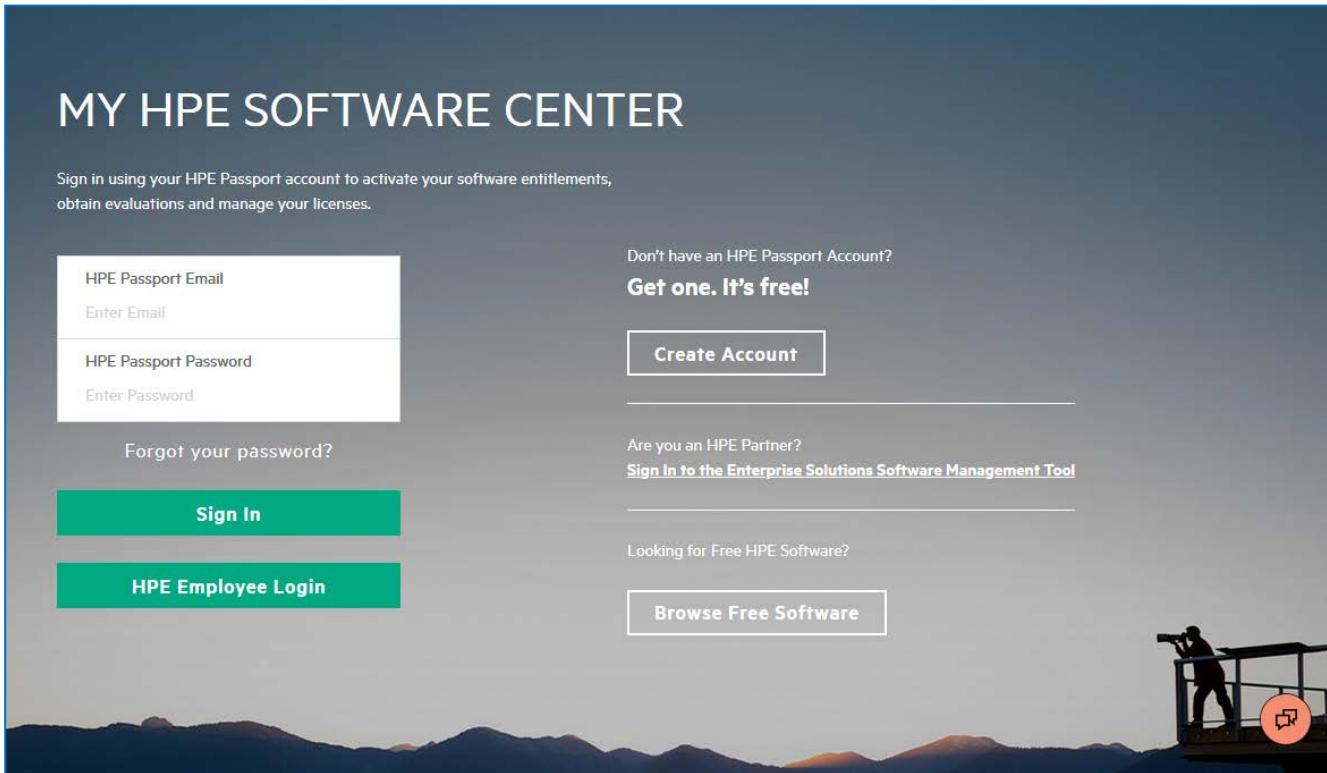


The screenshot shows the 'Partner Ready Portal' login interface. At the top left is the 'Hewlett Packard Enterprise' logo. A dropdown menu at the top right shows 'English UK'. Below the header is a large banner image of two men in a server room. To the right of the banner is a 'Passport Sign In' form with fields for 'Email Address' and 'Password', and a 'SIGN IN' button. Below the sign-in form are links for 'New user? Register here' and 'Forgot User ID or Password?'. At the bottom of the page, there are two sections: 'An easier way to engage' (describing the portal's purpose) and 'Want to become an HPE Partner?' (with a 'Get started and sign up today' link).

- 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_5.00_Synergy_ESXi_Z7550-96681.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' download 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 house icon, user profile, globe, and help icons. The main title is 'HPE OneView Demonstration Appliance'. Underneath the title is a large green 'Download' button. To the left of the button is a text box containing the following description: '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 the 'Download' button 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_5.20_Synergy_ESXi_Z7550-96879.ova** then click **Download**

Product Family: HPE OneView

Download Files View Activation Details

Download Files

Software (24)

HPE_ONEVIEW_DCS_5.20_SYNERGY_ESXI_Z7550-96879.OVA (2.22 GB)
SHA256 Checksum: f8e32b3733c26719eff9935b46ad96b1b28d78a37893034821...
[\(Copy\)](#)

HPE_ONEVIEW_DCS_5.20_SYNERGY_HYPER_V_Z7550-96880.ZIP (1.93 GB)
SHA256 Checksum: 1a6936f12dd2f271818730a0e7ee3547f7e747b82be9e2eba8...
[\(Copy\)](#)

HPE_ONEVIEW_DCS_5.20_SYNERGY_KVM_Z7550-96881.TAR.GZ (2.15 GB)
SHA256 Checksum: 1f4797188074d3bf4b3ec5b3228a0615967793678075b69915...
[\(Copy\)](#)

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

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

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

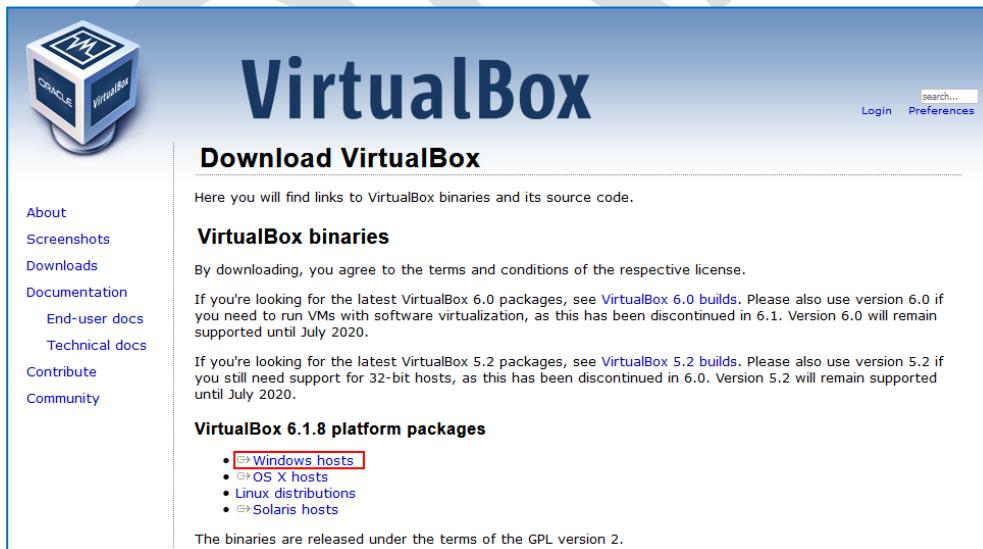
Downloading and installing VirtualBox on your PC

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).

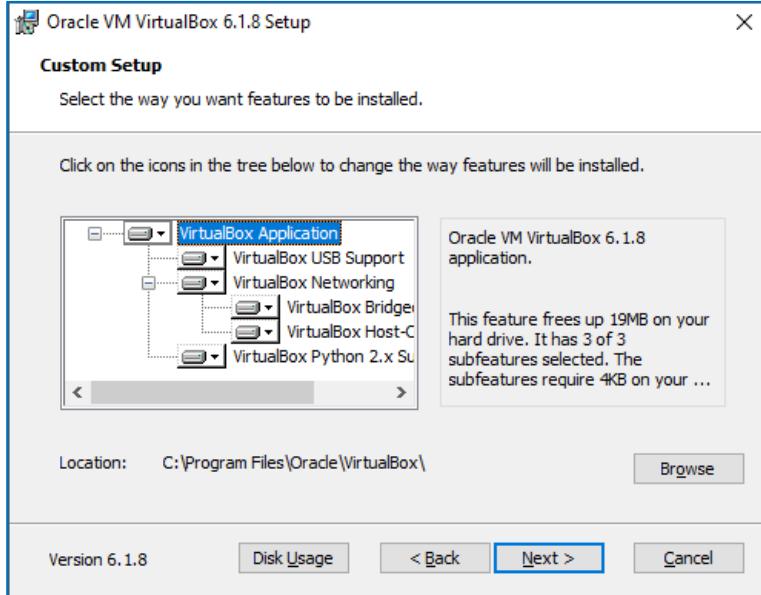
As mentioned in the prerequisites, the HPE OneView demonstration appliance is not officially supported with a VirtualBox hypervisor. The main reason for that is simply because it has not been tested and validated by HPE, but the experience shows that everything is working properly and as expected if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

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

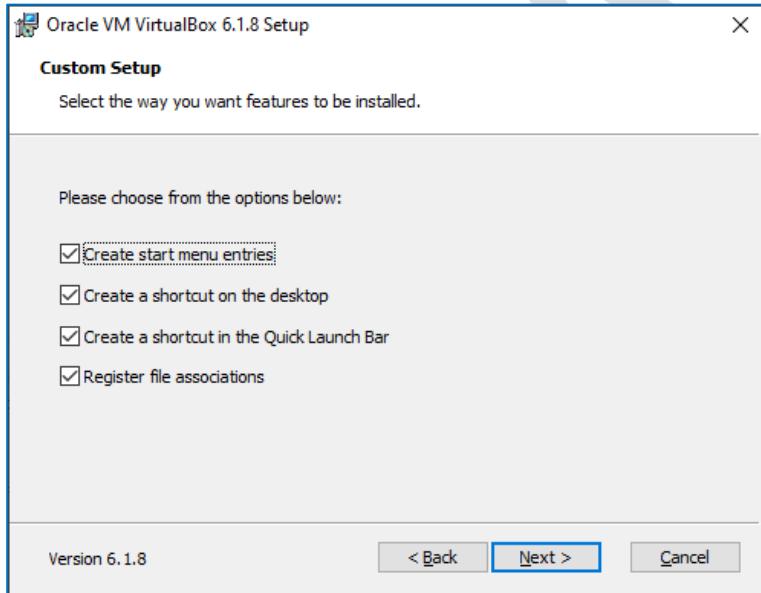


- Save the file on your PC and launch the executable.

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



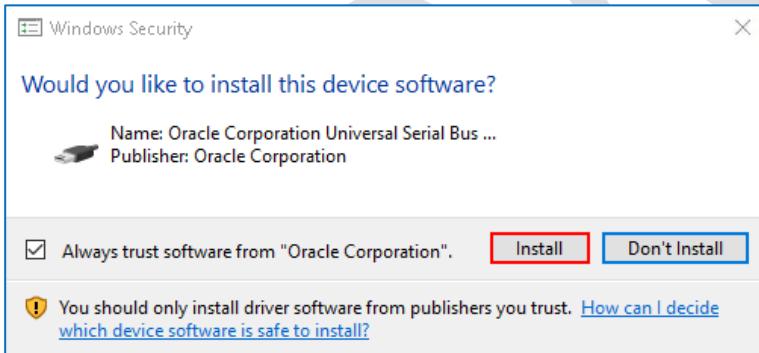
- Click **Next**



- Click **Next**



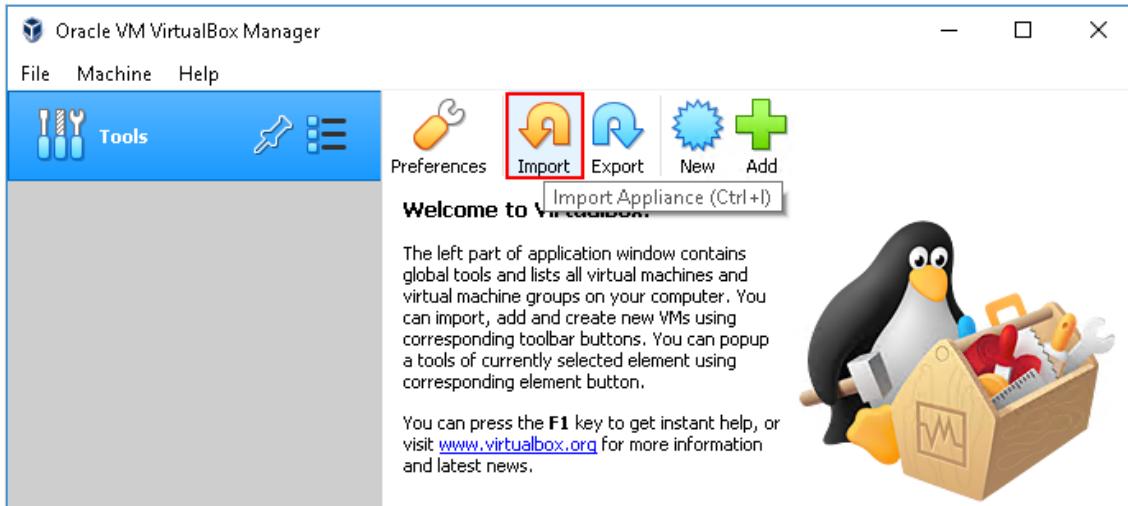
- Then click **Yes** to proceed with the installation then click **Install**
- When the Windows Security message pops-up, click **Install**



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

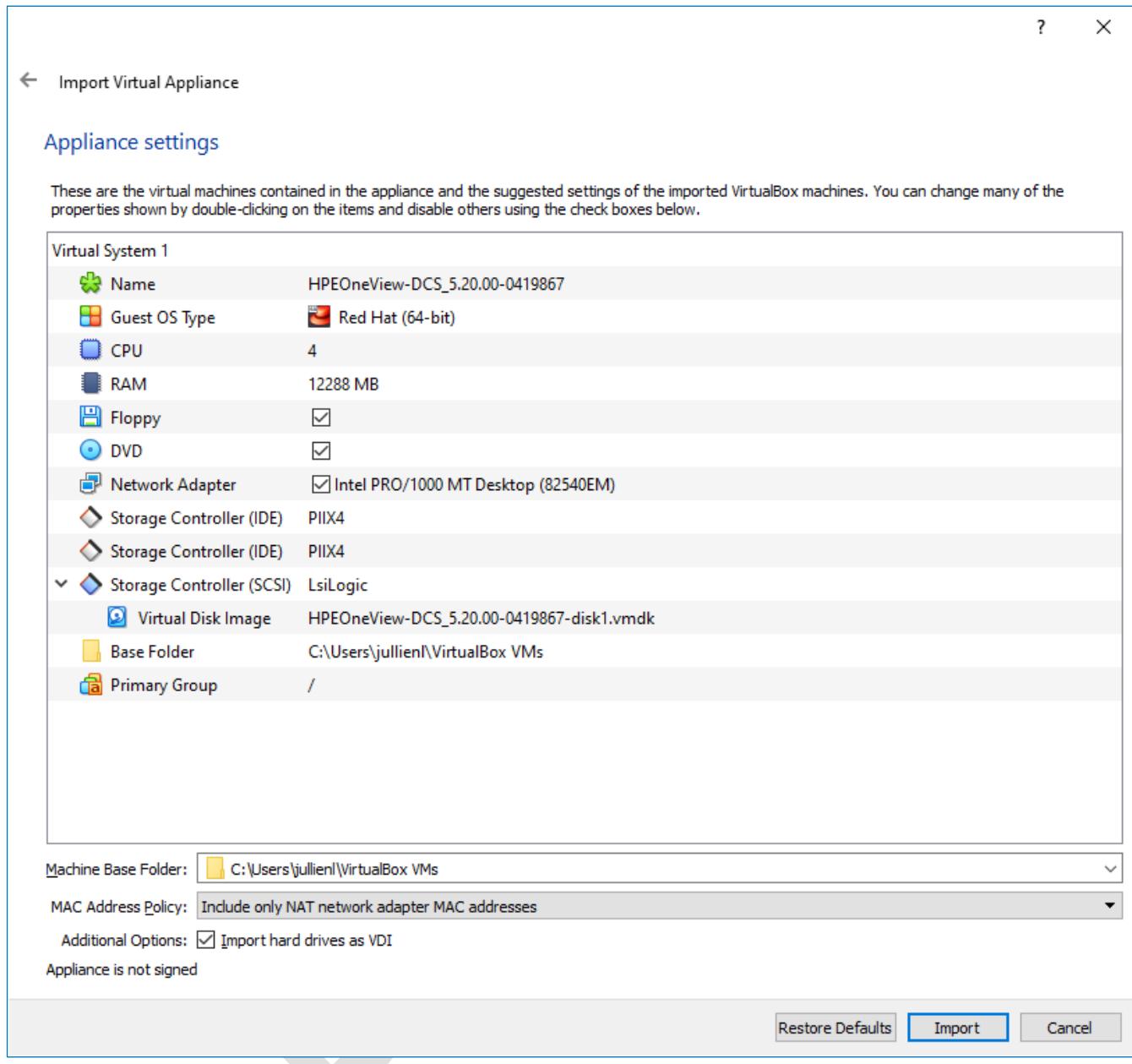
Importing and tuning the HPE OneView Demonstration appliance in VirtualBox

- In VirtualBox, select **Import**

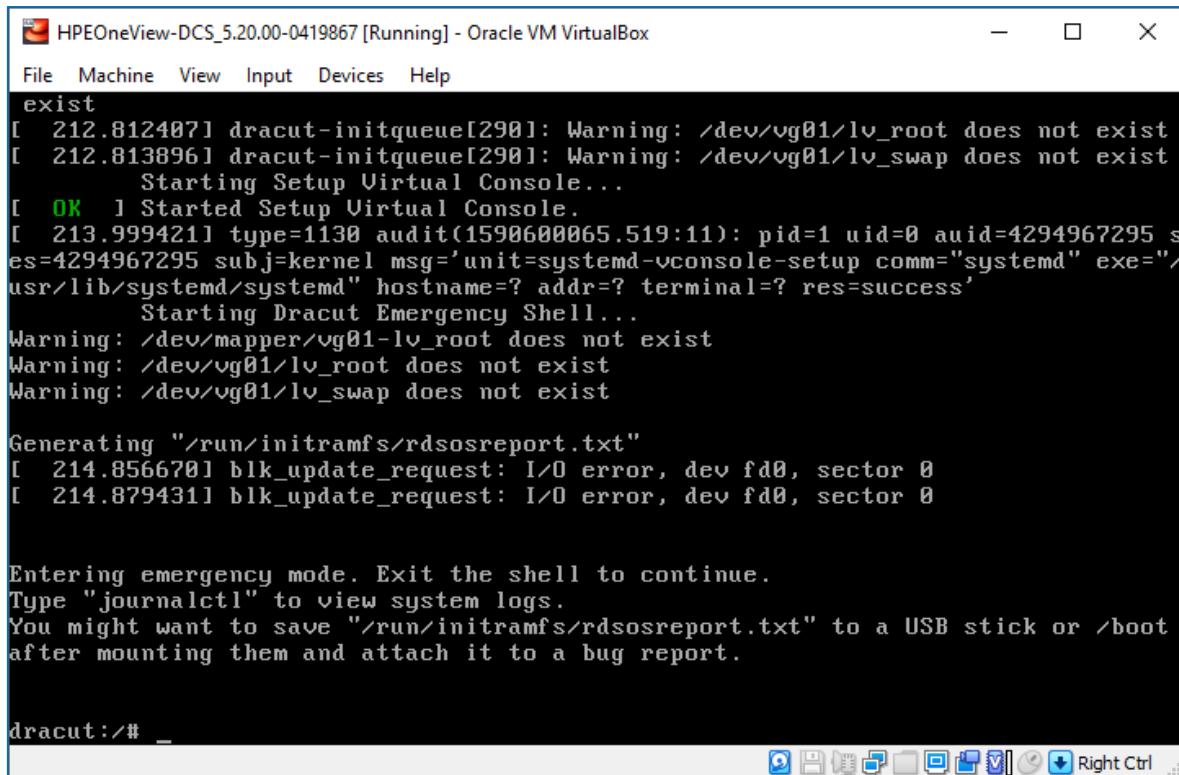


- Select the **HPE_OneView_DCS_5.20_Synergy_ESXi_Z7550-96879.ova** file

- 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

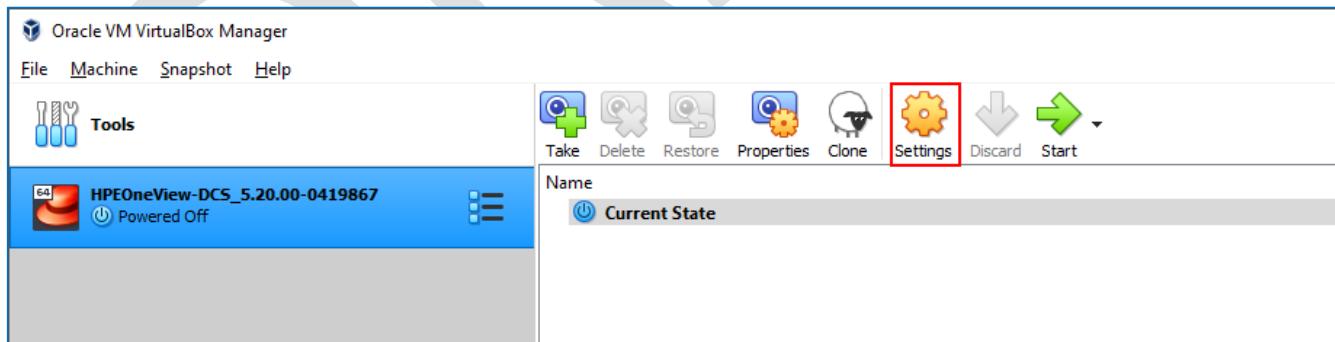
Generating "/run/initramfs/rdsosreport.txt"
[ 214.856670] blk_update_request: I/O error, dev fd0, sector 0
[ 214.879431] blk_update_request: I/O error, dev fd0, sector 0

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.

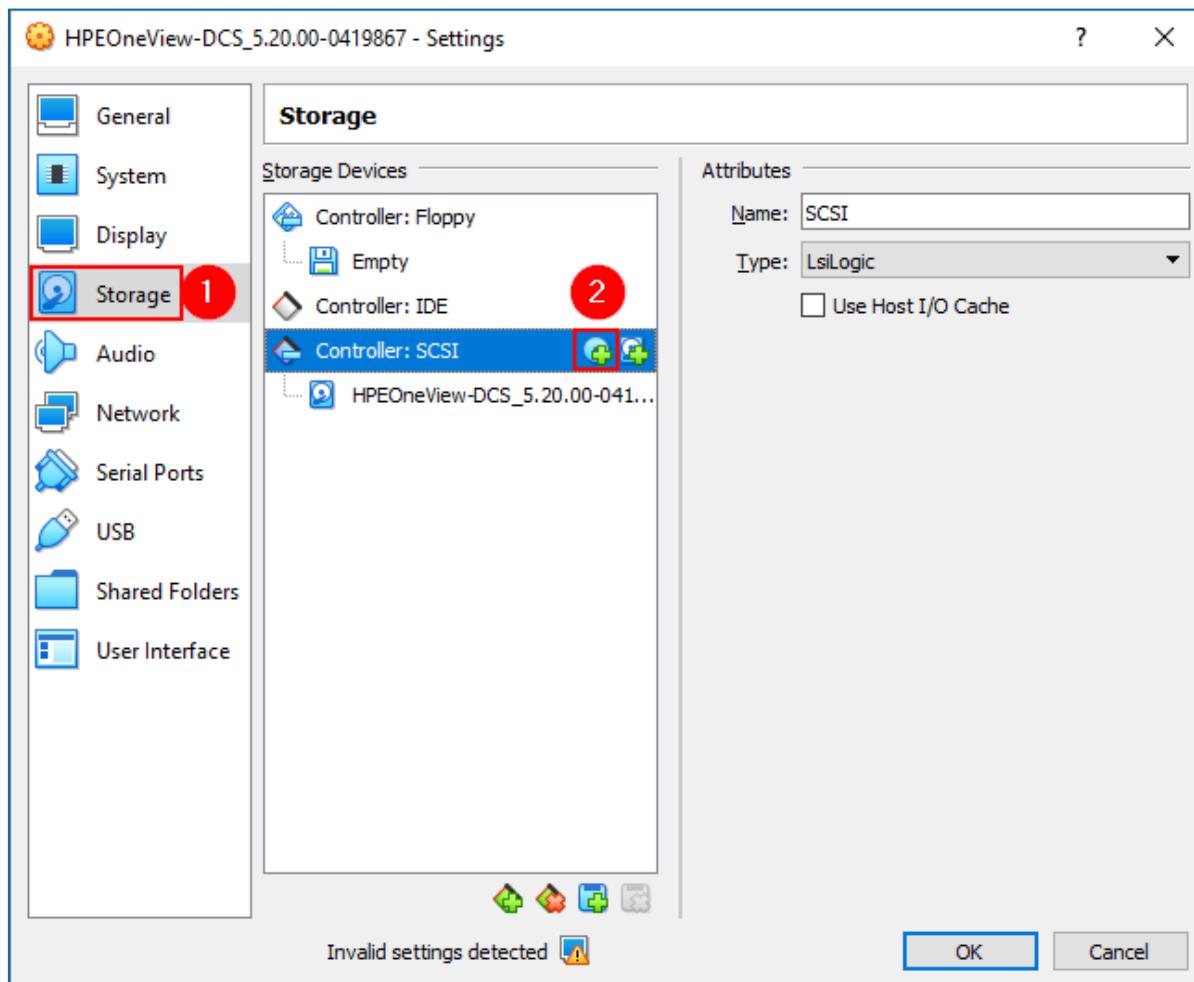
dracut:/# _
```

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

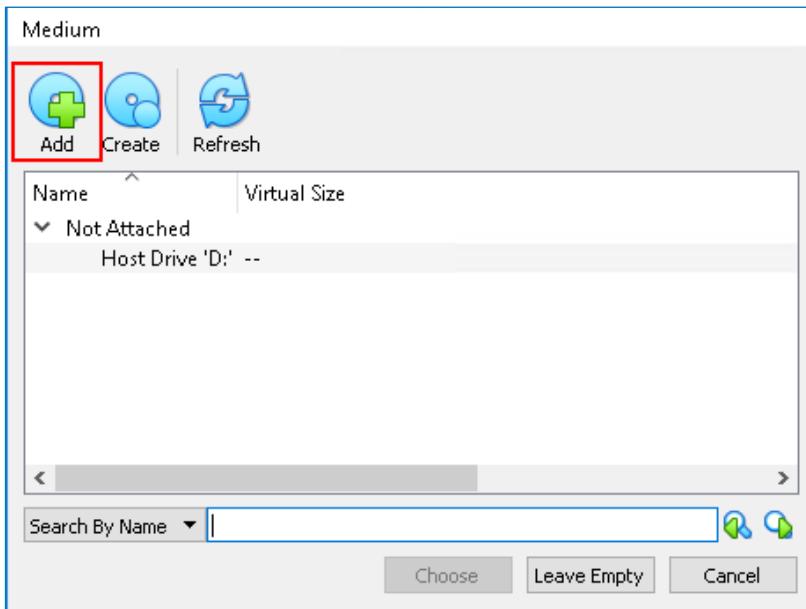
- Edit the VM Settings



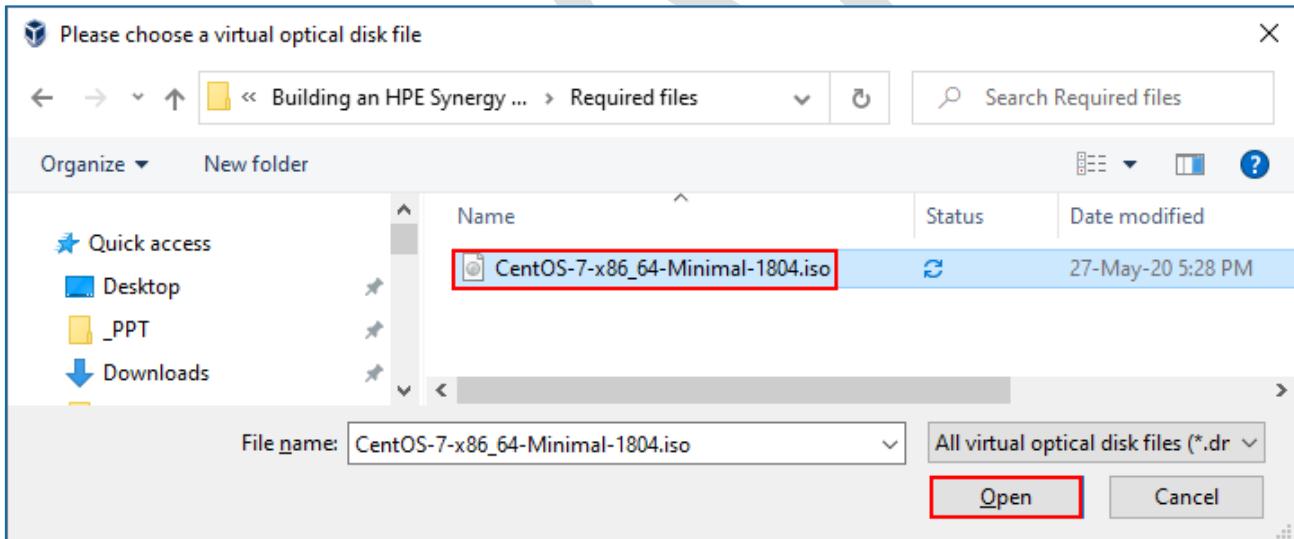
- Select **Storage** then click on **Adds optical drive** icon on the iSCSI Controller



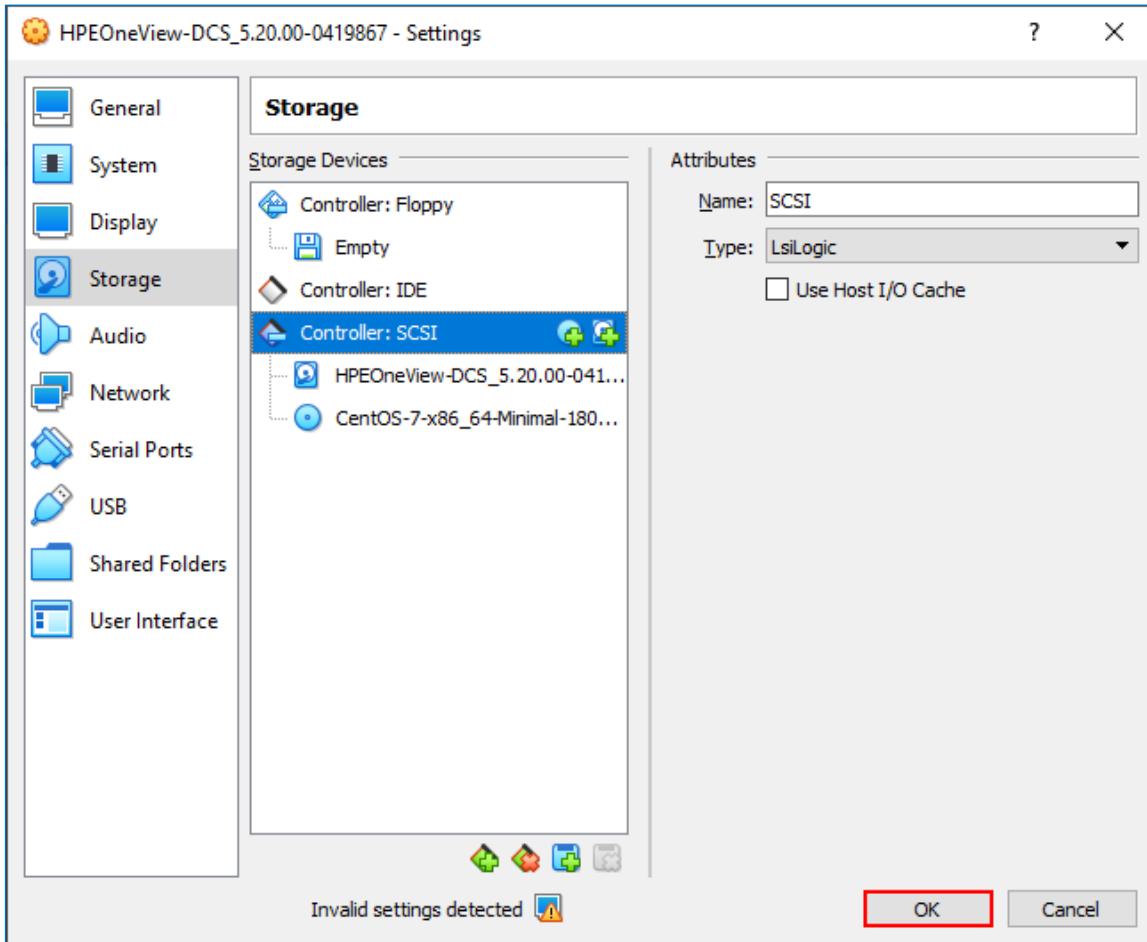
- Select **Add**



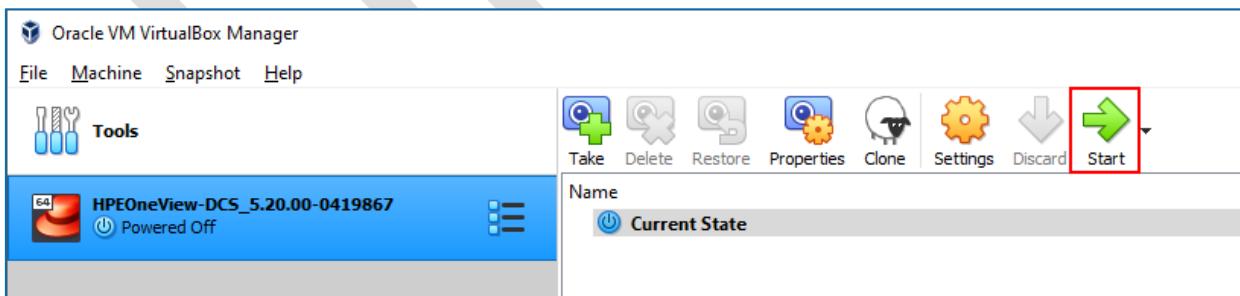
- Select the CentOS ISO image then click **Open**



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



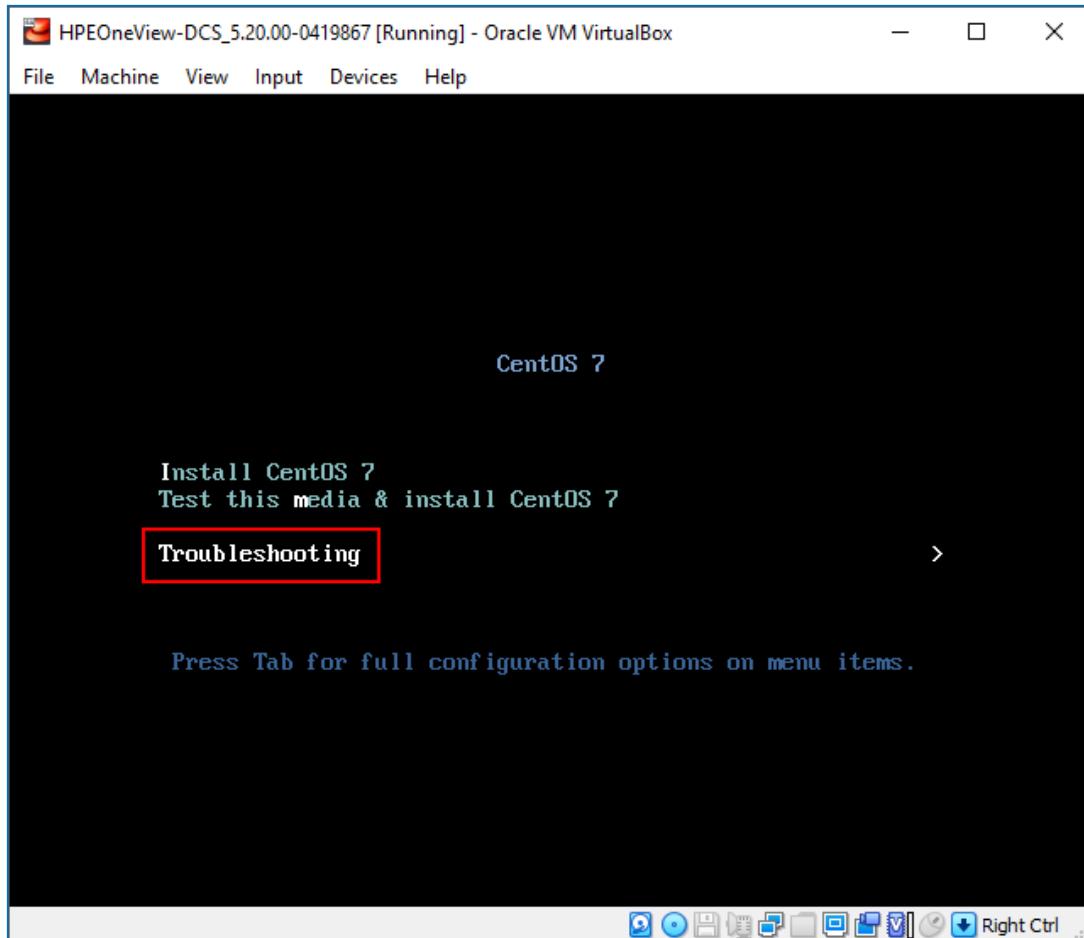
- Then start the VM by pressing **Start**



The CentOS starting menu should come up.

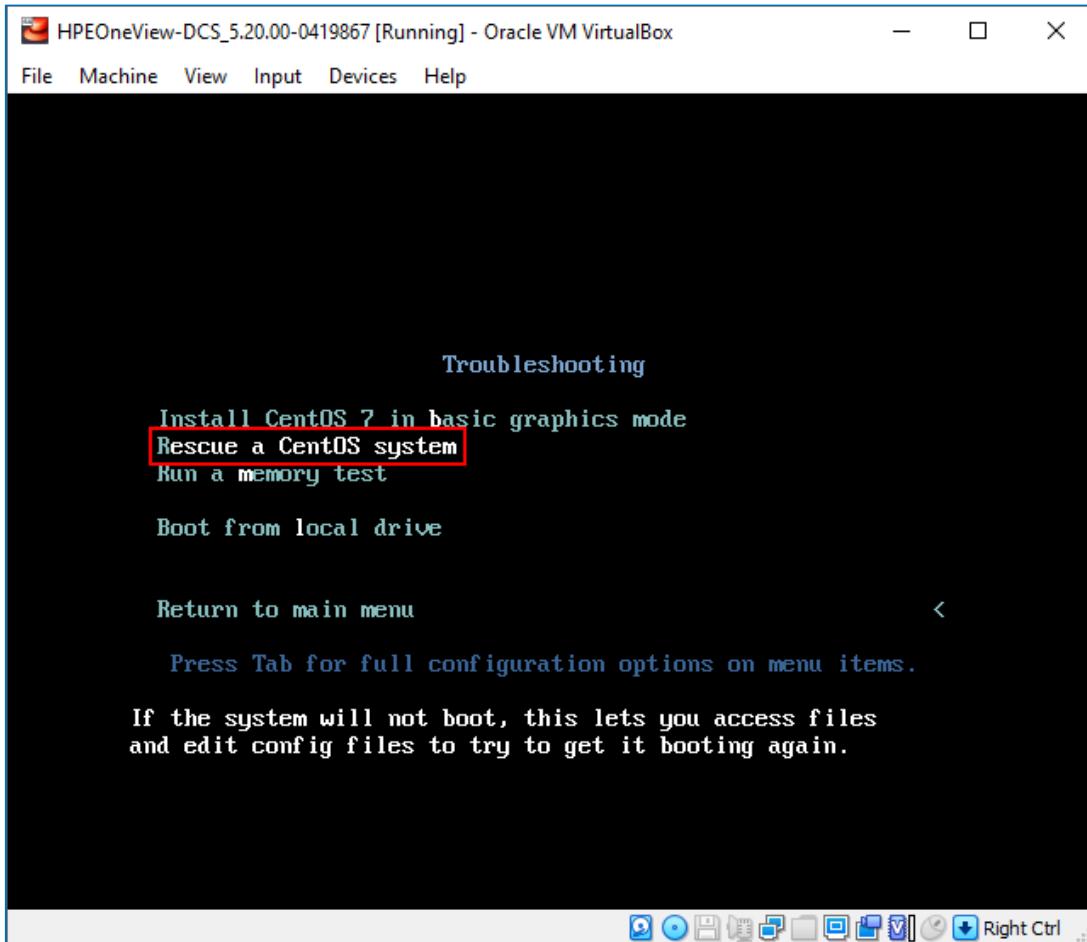
Note: If you are seeing some errors when you start the VM, go to the [Troubleshooting](#) section

- Select **Troubleshooting**

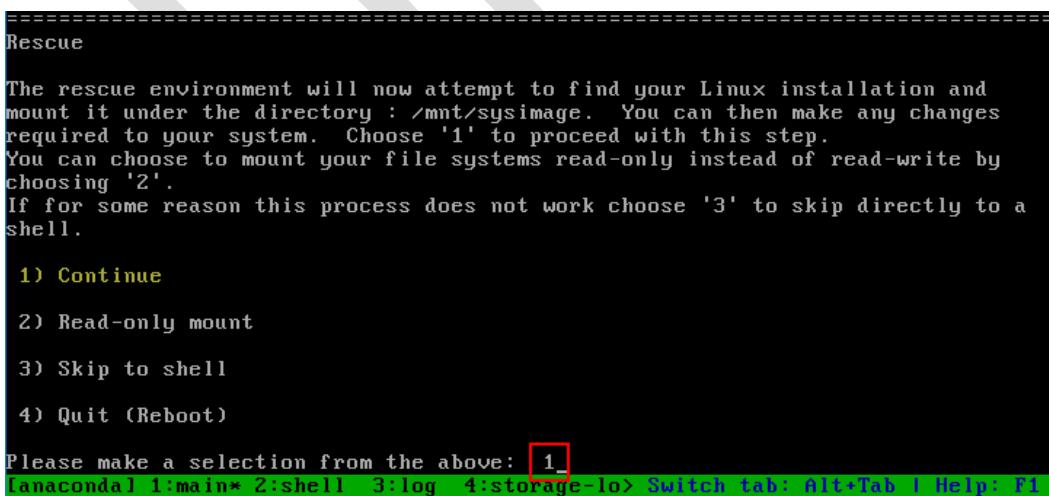


Note: To release the mouse from a VirtualBox console use the right Control key on your keyboard

- 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 5.20 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
```

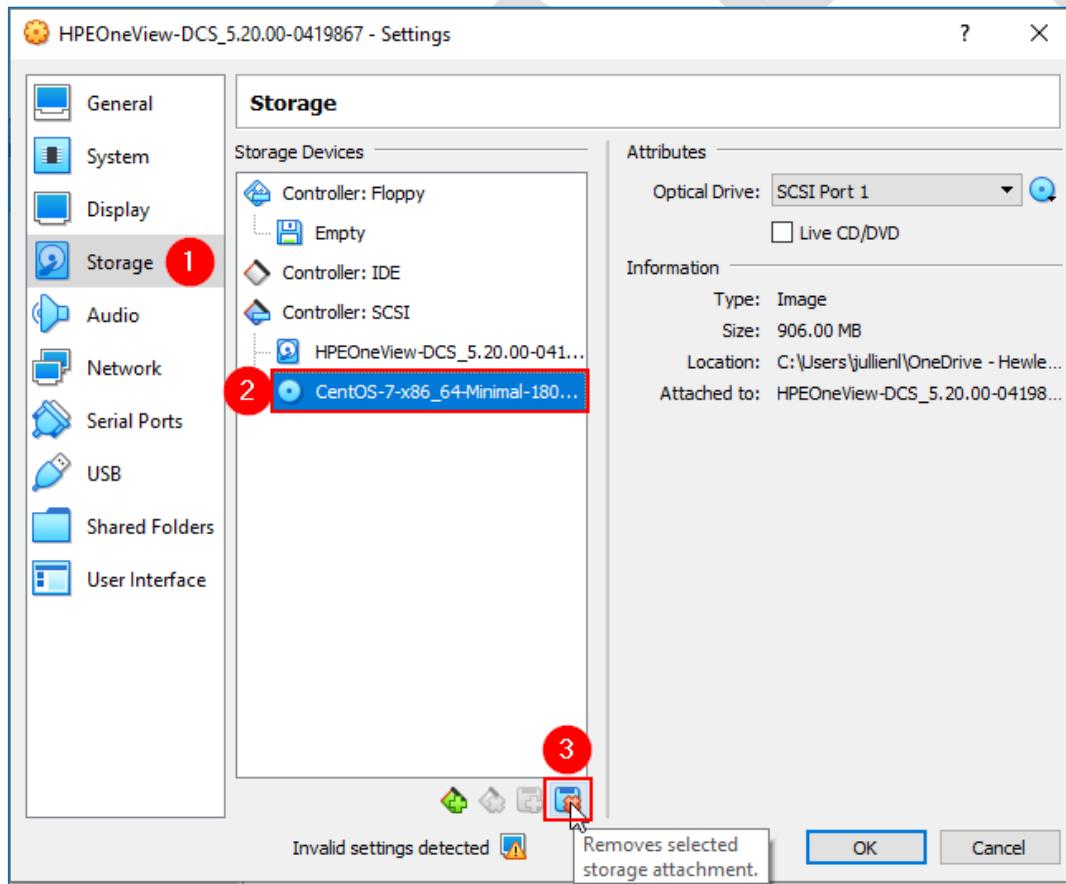
The command prompt should change from *bash* back to *sh*.

- Then shut down the VM:

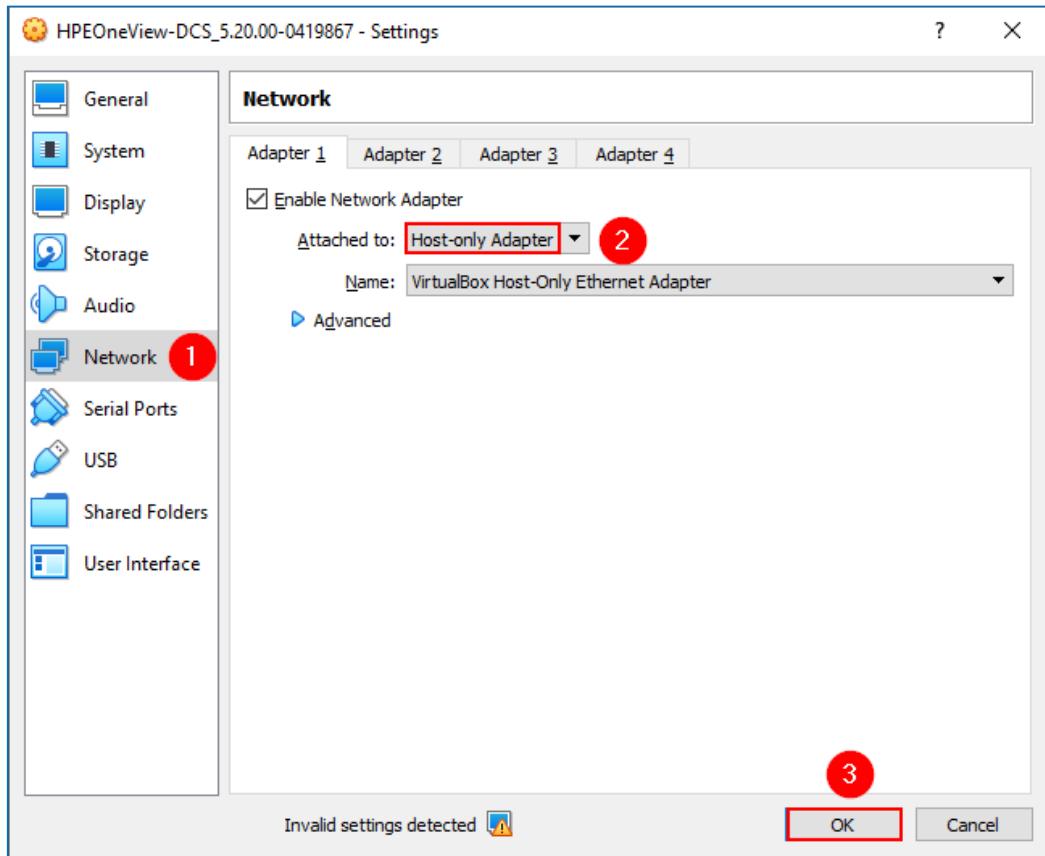
```
shutdown now
```

```
[ 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...LGNt6qt19MViAJSQ9Mm0de0TjV8hxJi.
[ 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.



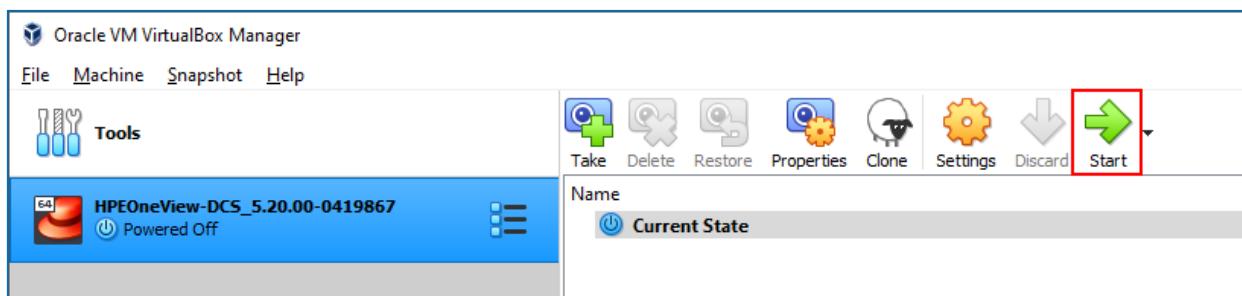
- Then we need to configure the VM network to use **Host-only Adapter** because the appliance requires the network NOT to use DHCP:



- Click **OK** to save changes.
- Shutdown all applications running on your computer before initiating the startup process. This relieves memory/CPU/disk pressure on the VM while booting.

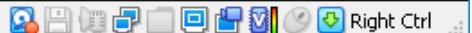
Important notice: It is recommended that you close all programs before starting the DCS appliance as the lack of CPU/memory resources during startup and hardware discovery can negatively impact the appliance. Once the appliance is started and the initial discovery is complete, the lack of resources is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

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

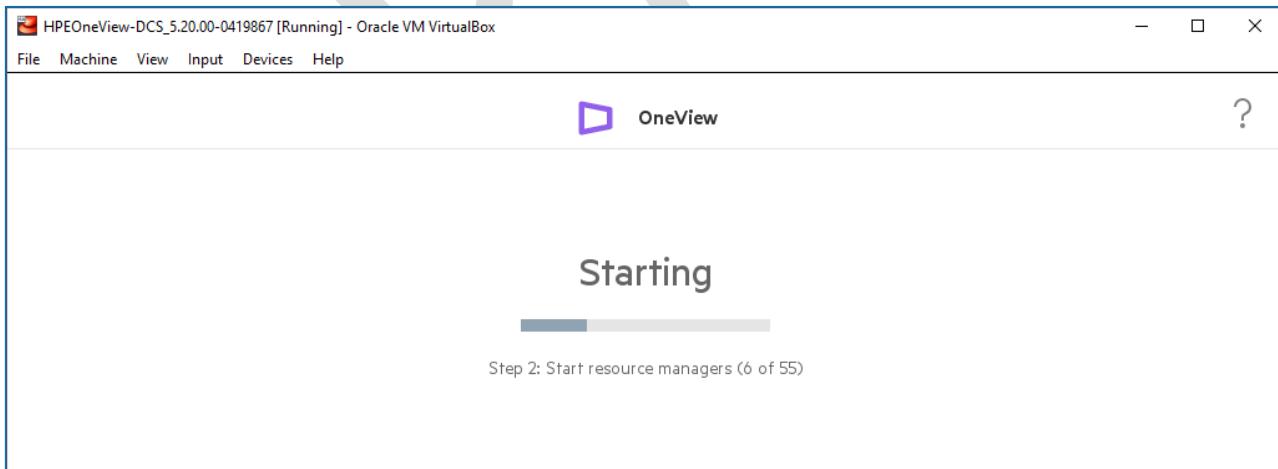


- At this point it should begin its normal unpacking process which may reboot the VM a few times.

```
[ 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.

DRAFT



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).

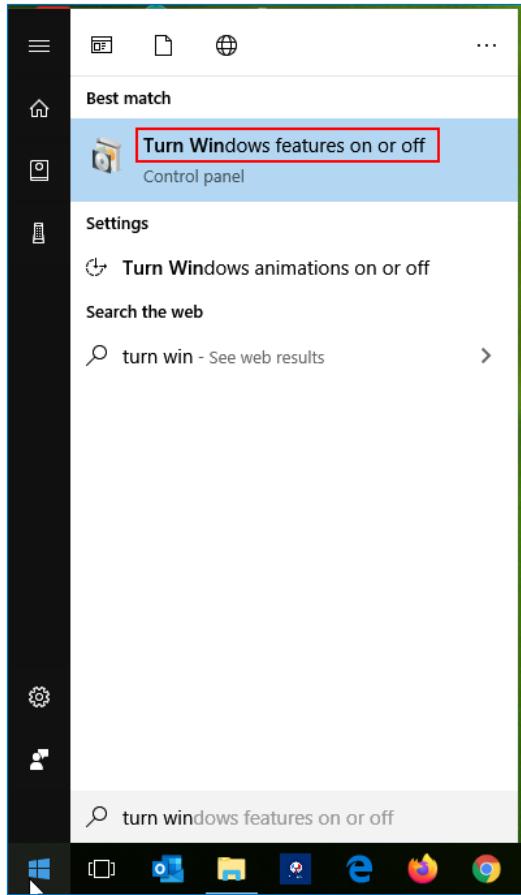
Note: The old version of WSL in Windows version 1709 can also be used for this lab.

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>.

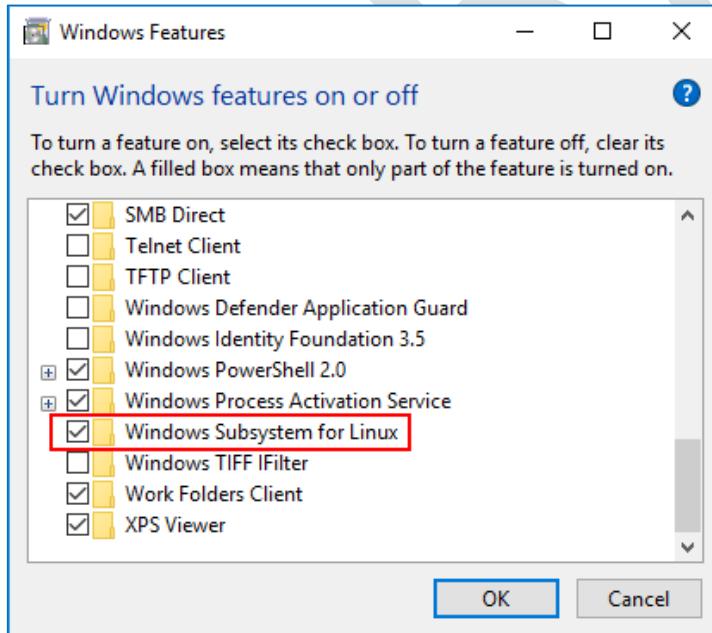
WSL 2 is only available in Windows 10, Version 2004, Build 19041 or higher.

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.



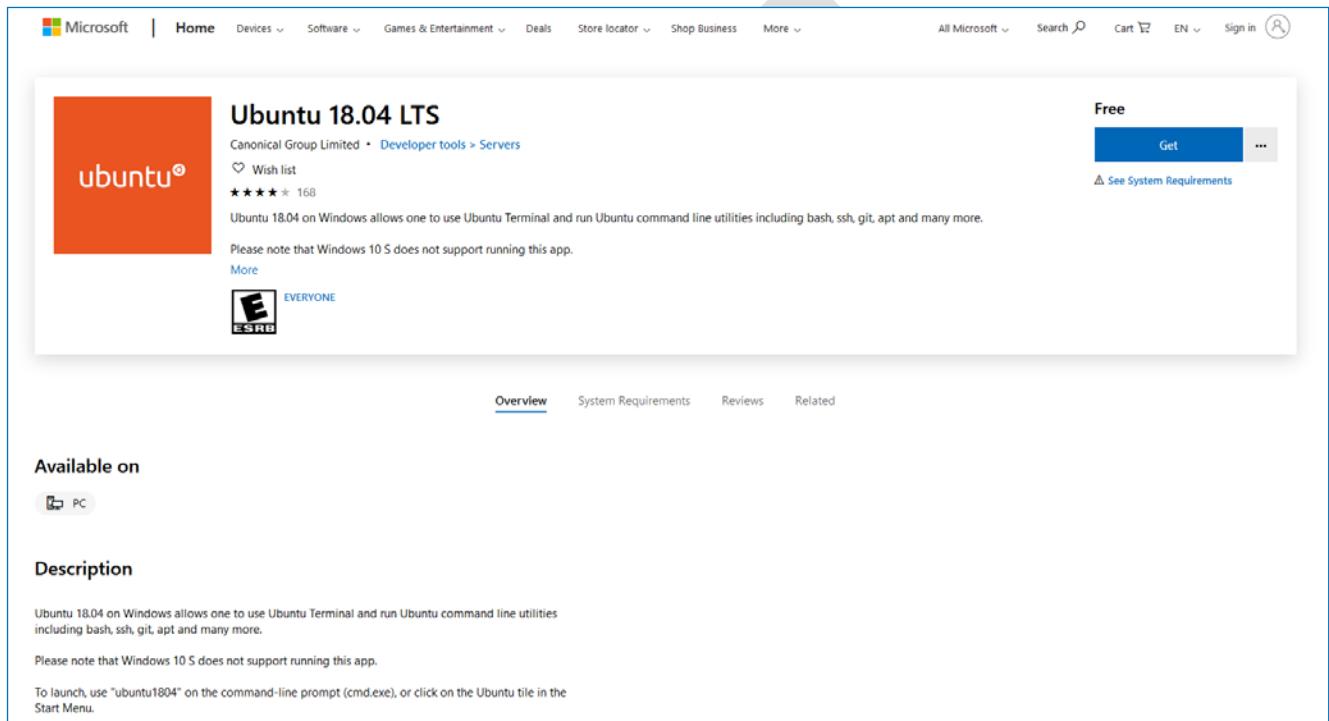
- Once turned on, follow the instructions, and do any restarts if you have to.

Installing Ubuntu Windows Subsystem for Linux

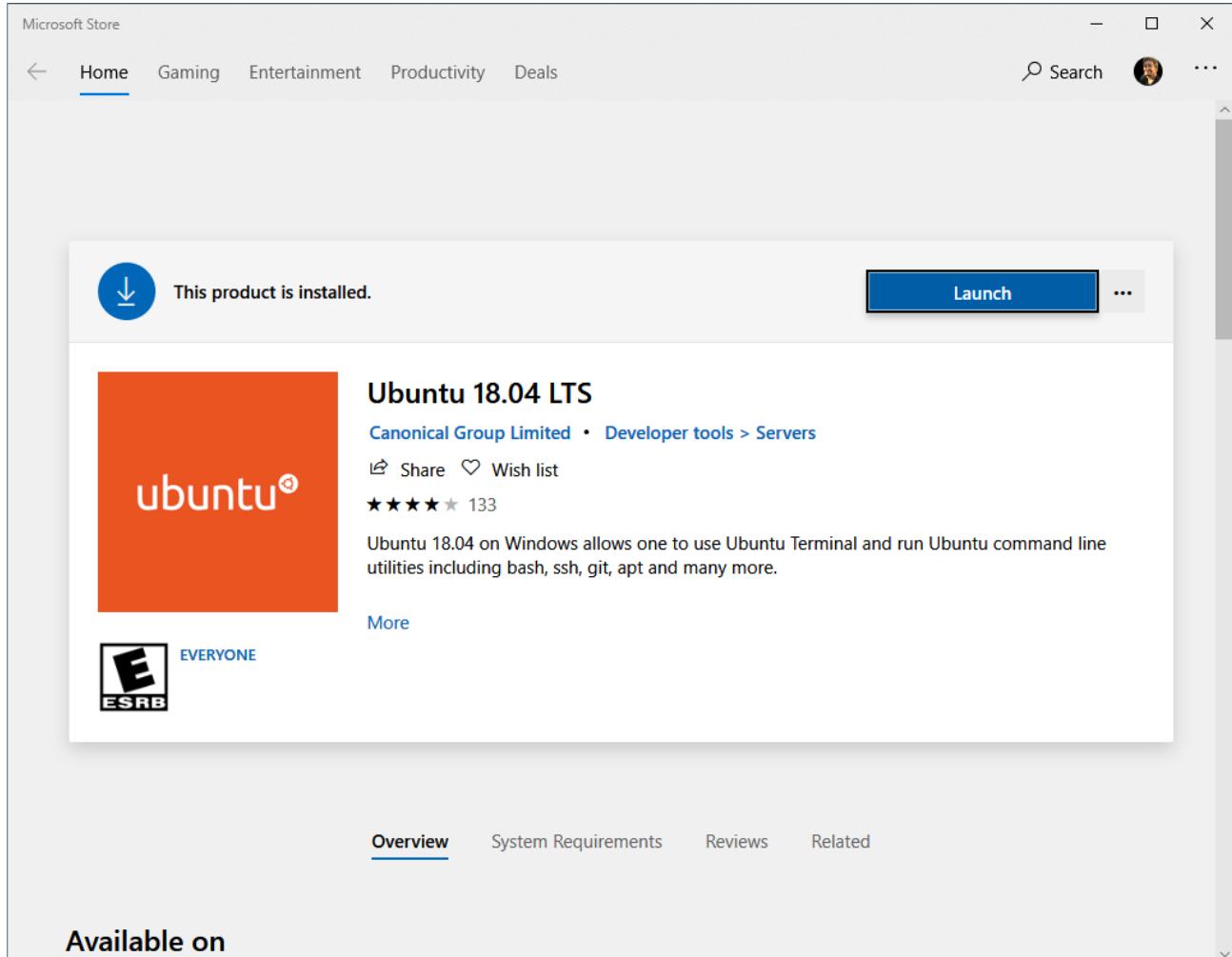
Next, we need to install a Linux distribution, we are going to use Ubuntu since that is one of the most popular.

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>

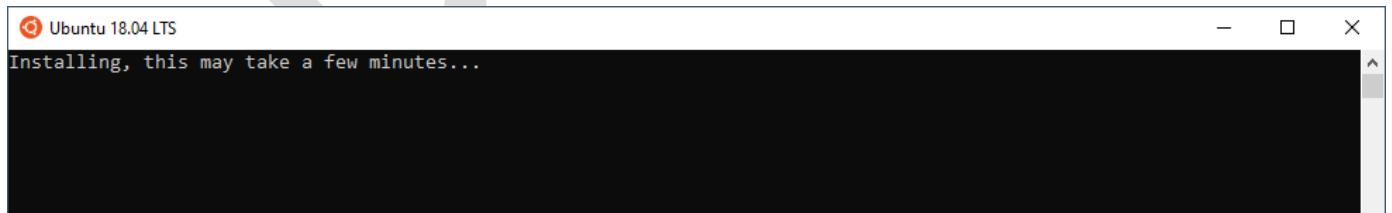
- To install Ubuntu 18.04 LTS, open the Microsoft Store at <https://www.microsoft.com/en-us/p/ubuntu-1804-lts/9n9tngvndl3q>
- Then select **Get**



- 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.

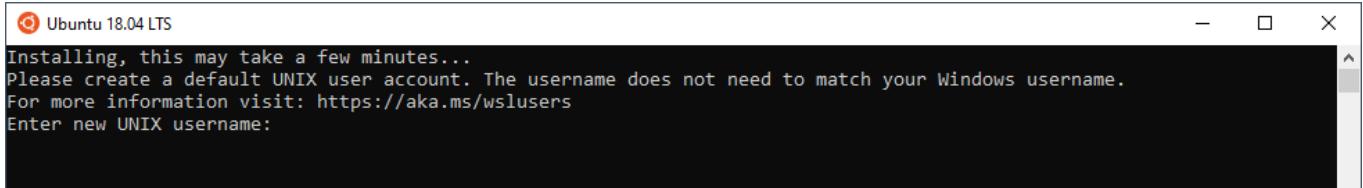


- The first time a newly installed distro runs, a Console window will open, and we will be asked to wait for a minute or two for the installation to complete.



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).

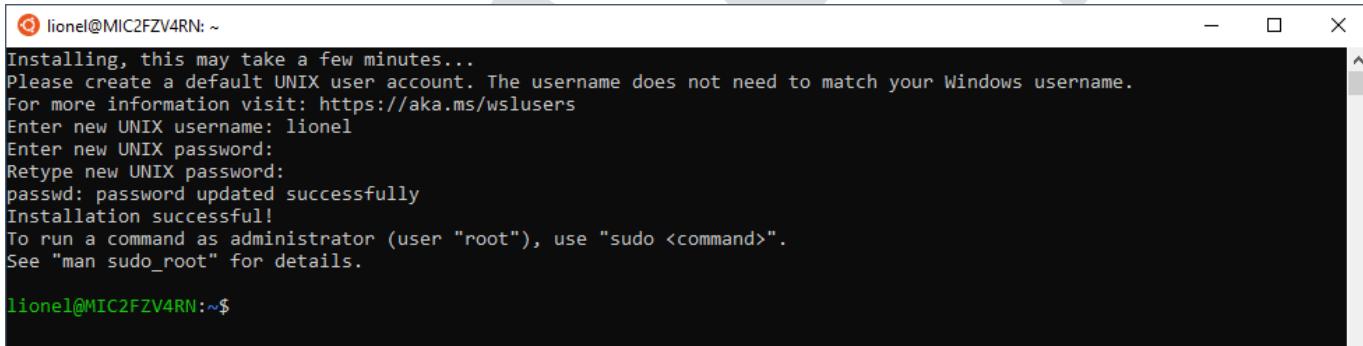


```
Ubuntu 18.04 LTS
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

- 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!

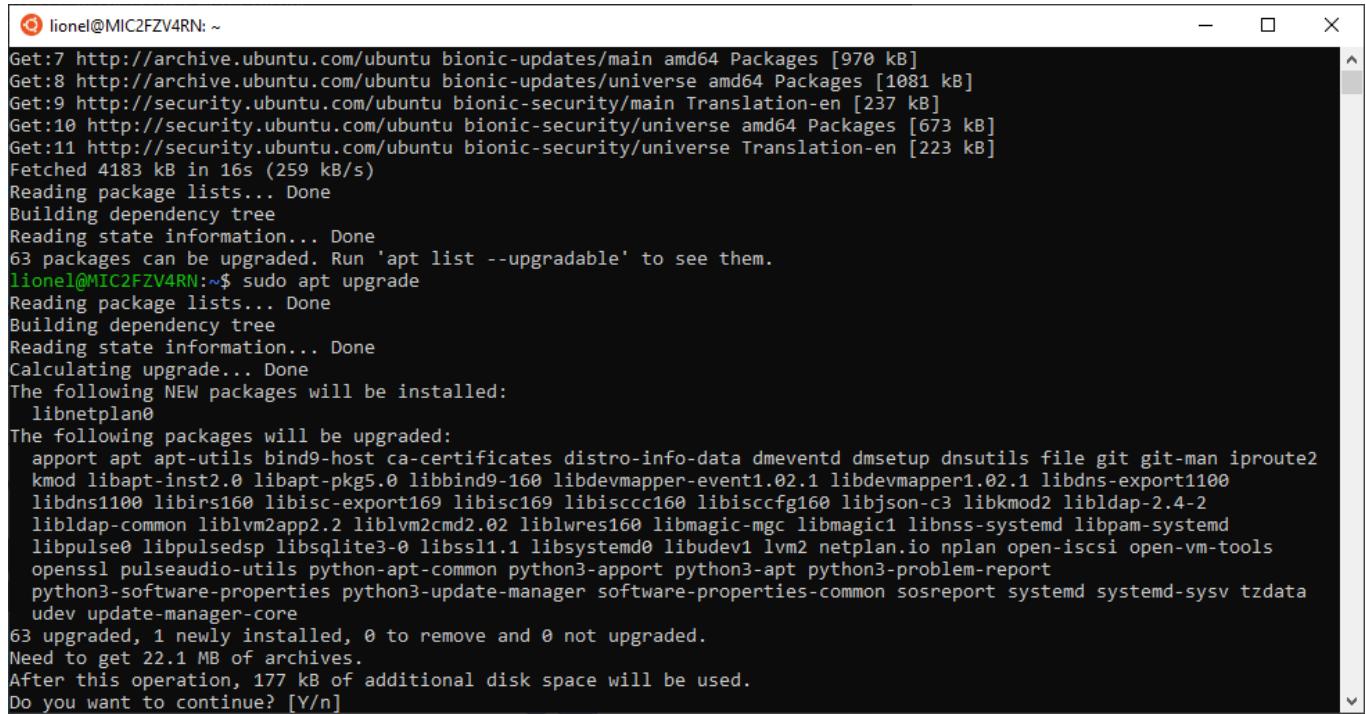


```
lionel@MIC2FZV4RN: ~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: lionel
Enter new UNIX password:
Retype new UNIX password:
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
```



A screenshot of a Windows terminal window titled "lionel@MIC2FZV4RN: ~". The window displays the output of the "apt update" and "apt upgrade" commands. The output shows various package downloads and upgrades, including the installation of libnetplan0 and the upgrade of numerous other packages like libapt-inst2.0, libbind9-160, and libdevmapper-event1.02.1. It also shows the user being prompted to continue the operation.

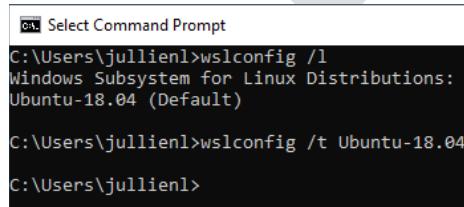
```
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [970 kB]  
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1081 kB]  
Get:9 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [237 kB]  
Get:10 http://security.ubuntu.com/ubuntu 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.  
lionel@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.



A screenshot of a Windows Command Prompt window titled "Select Command Prompt". The window shows the execution of the "wslconfig /l" command, which lists the available Linux distributions: "Windows Subsystem for Linux Distributions: Ubuntu-18.04 (Default)". It then shows the execution of the "wslconfig /t Ubuntu-18.04" command, which sets the default distribution to Ubuntu-18.04.

```
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:

The screenshot shows the GitHub README.md page for the 'oneview-ansible' repository. It includes a 'build passing' badge and a 'coverage 100%' badge. The main content is titled 'Ansible Modules for HPE OneView' and describes it as 'Modules to manage HPE OneView using Ansible playbooks.' Below this is a 'Requirements' section with the following bullet points:

- 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
```

The screenshot shows a terminal window on a Linux system. The user 'lionel' is at the prompt. The command 'sudo apt-add-repository ppa:ansible/ansible' is entered, followed by a message from Ansible about its purpose. The user then presses [ENTER] to continue adding the repository.

```
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.

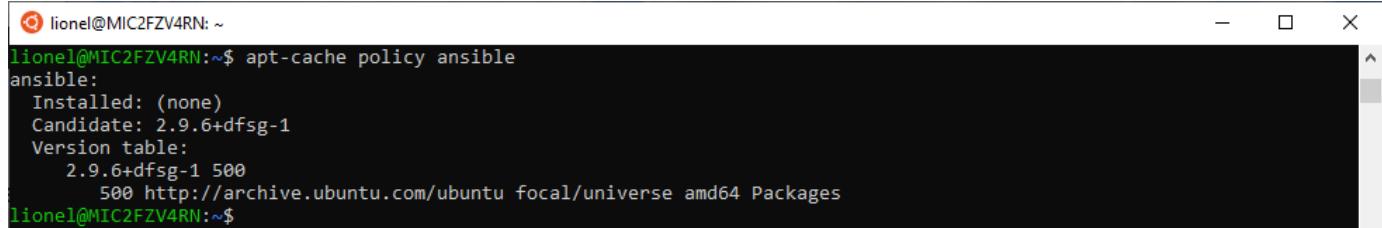
Note: if you are behind a corporate proxy, you need to set the proxy environment variables:

```
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
```

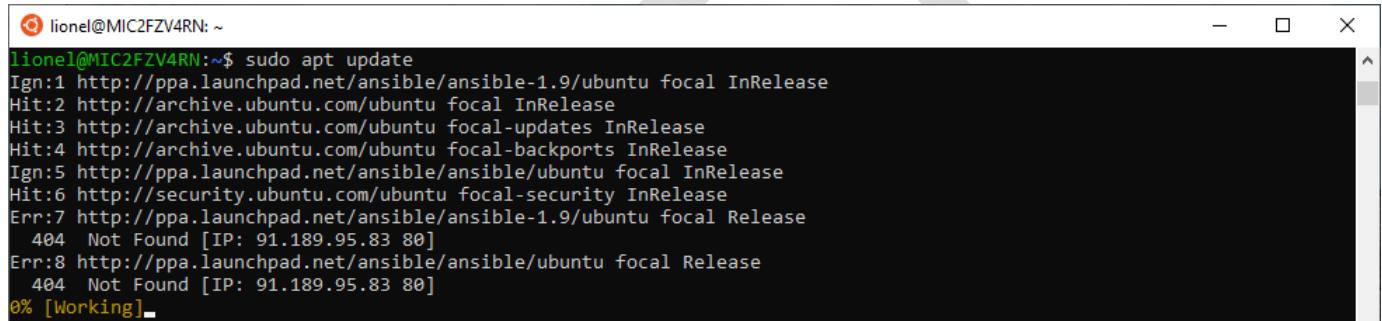


```
lionel@MIC2FZV4RN:~$ apt-cache policy ansible
ansible:
  Installed: (none)
  Candidate: 2.9.6+dfsg-1
  Version table:
    2.9.6+dfsg-1 500
      500 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages
lionel@MIC2FZV4RN:~$
```

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]
```

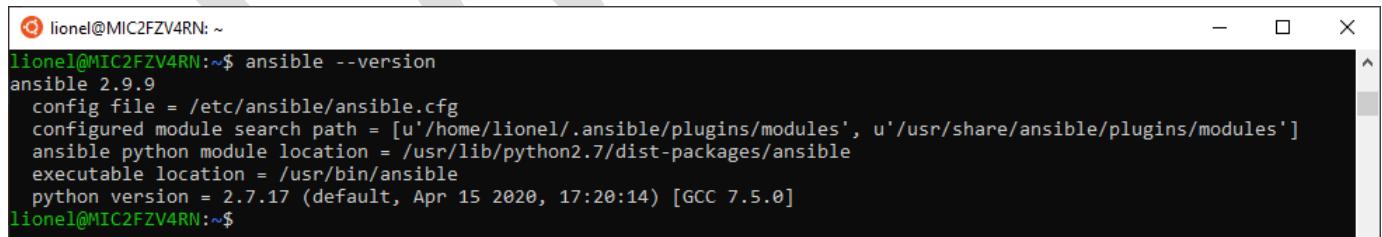
A terminal window titled 'lionel@MIC2FZV4RN:~' showing the output of the 'sudo apt update' command. It lists several package sources and their status, including hits for the local archive and errors for the Ansible PPA due to a 404 Not Found error.

- 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:~$
```

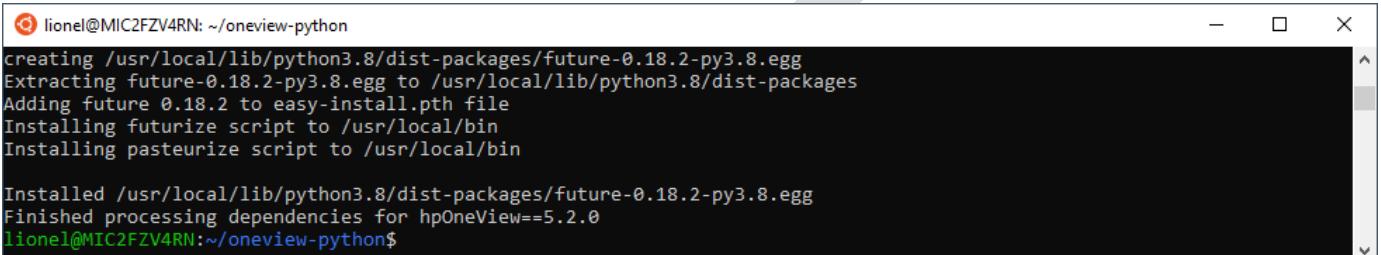
A terminal window titled 'lionel@MIC2FZV4RN:~' showing the output of the 'ansible --version' command. It displays detailed information about the installed Ansible version, configuration file, module search paths, and executable location.

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
```



```
lione1@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  
lione1@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

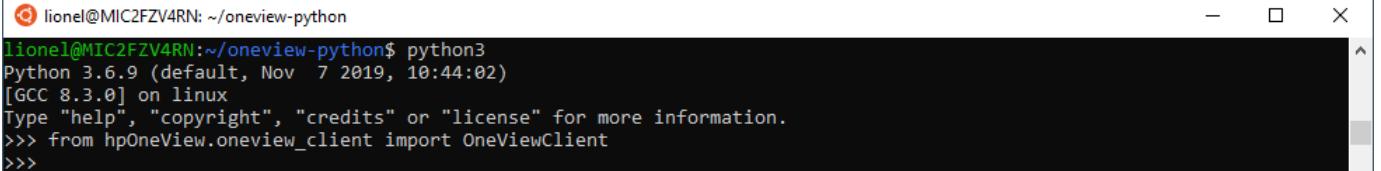
Note: if you are behind a corporate proxy, you need to set a Git proxy:
`git config --global http.proxy http://proxy.myproxy.com:port/
git config --global https.proxy http://proxy.myproxy.com:port/`

- 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
```



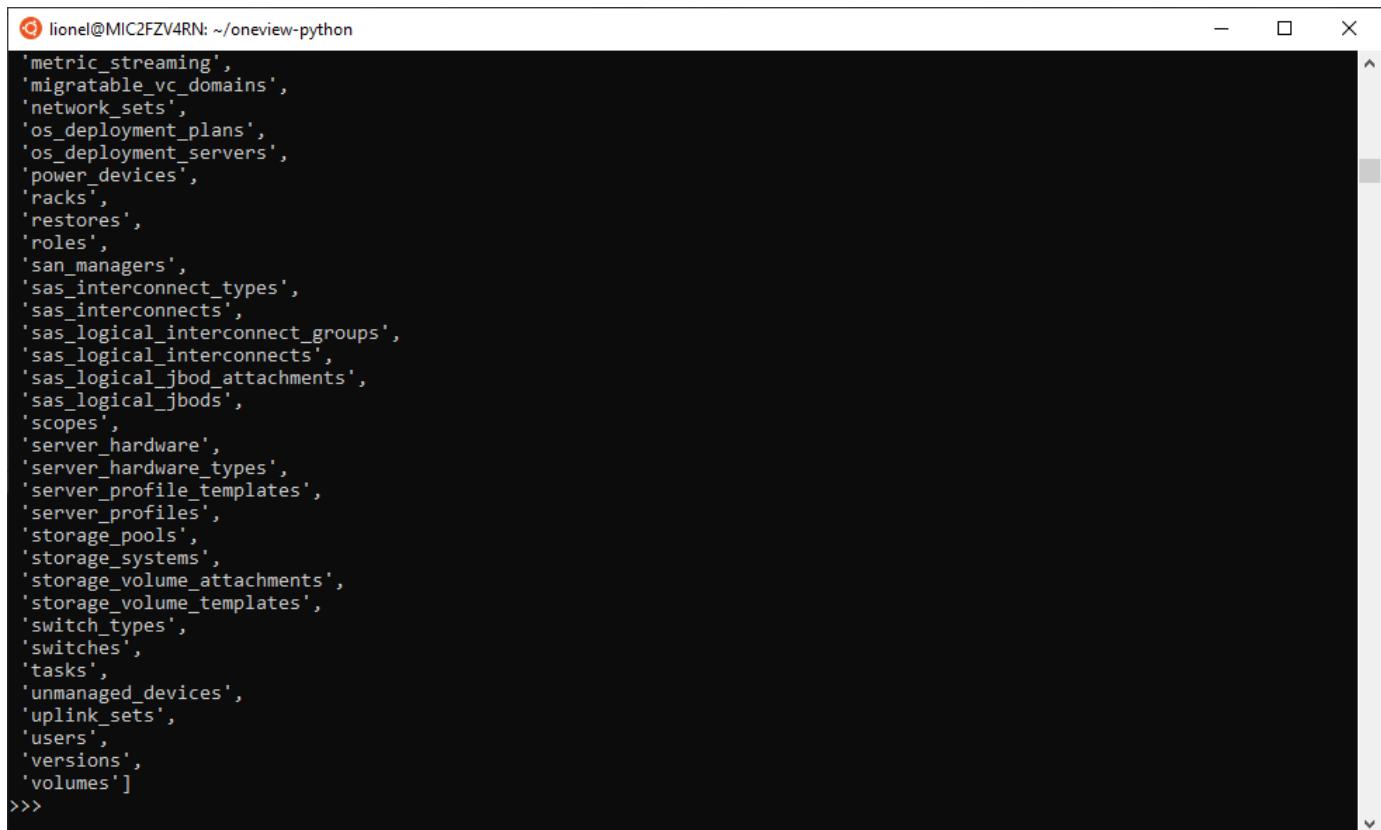
```
lione1@MIC2FZV4RN: ~/oneview-python$ python3  
Python 3.6.9 (default, Nov  7 2019, 10:44:02)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from hpOneView.oneview_client import OneViewClient  
>>>
```

- 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))
```



```
lione@MIC2FZV4RN: ~/oneview-python
['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',
 'volumes']
>>>
```

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
```

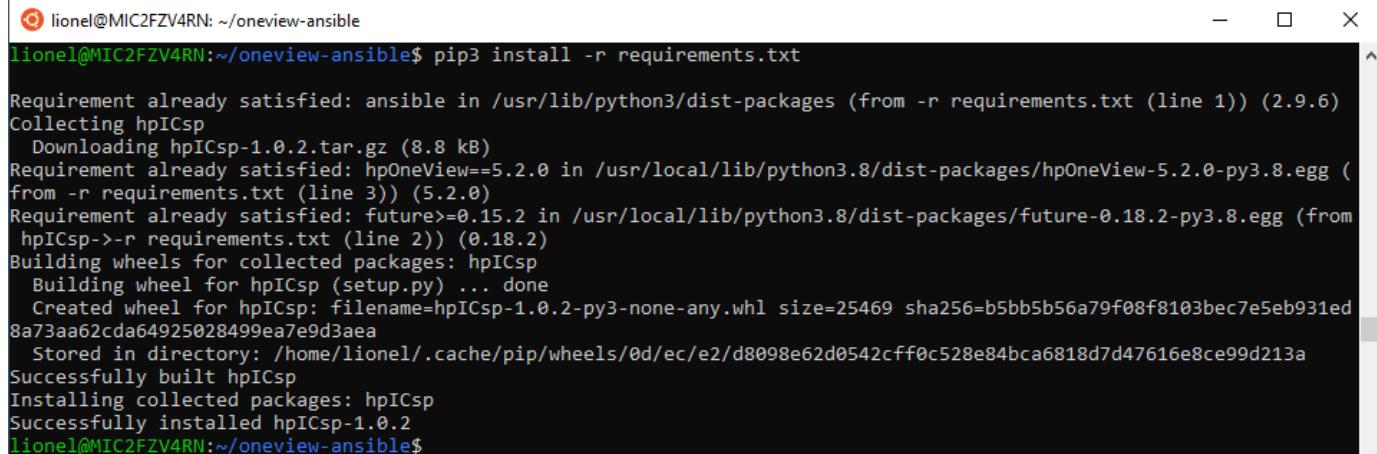
Note: if you are behind a corporate proxy, you need to set a Git proxy:

```
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-C 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.

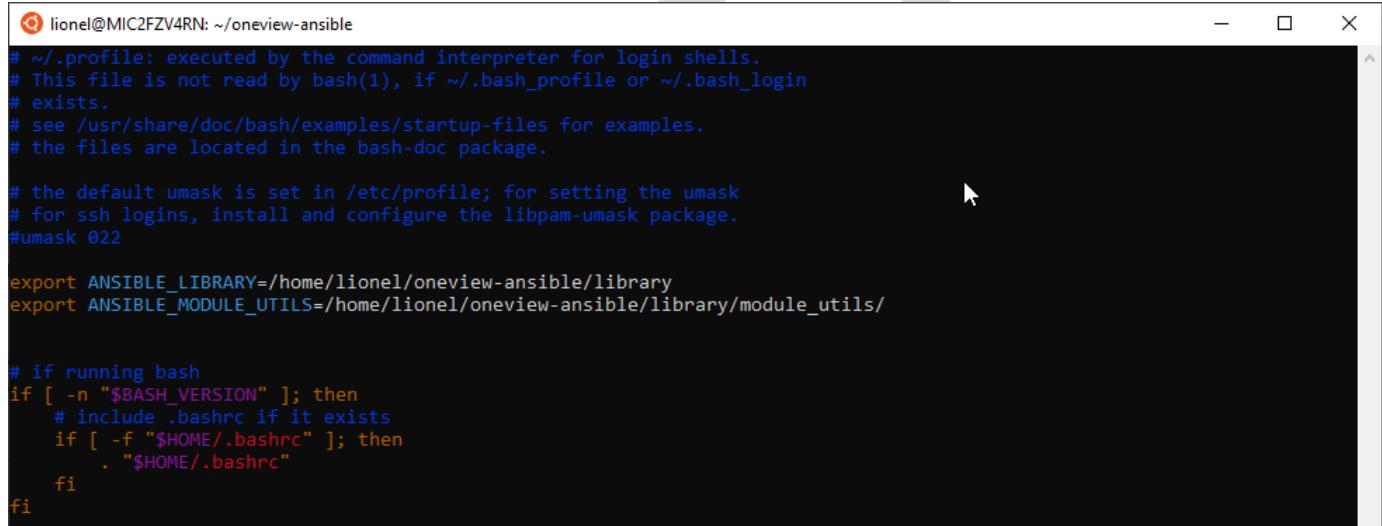
- Type:

```
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:*.lzha=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:*.bz2=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.zoo=01;31:*.cpio=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.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 ~
```

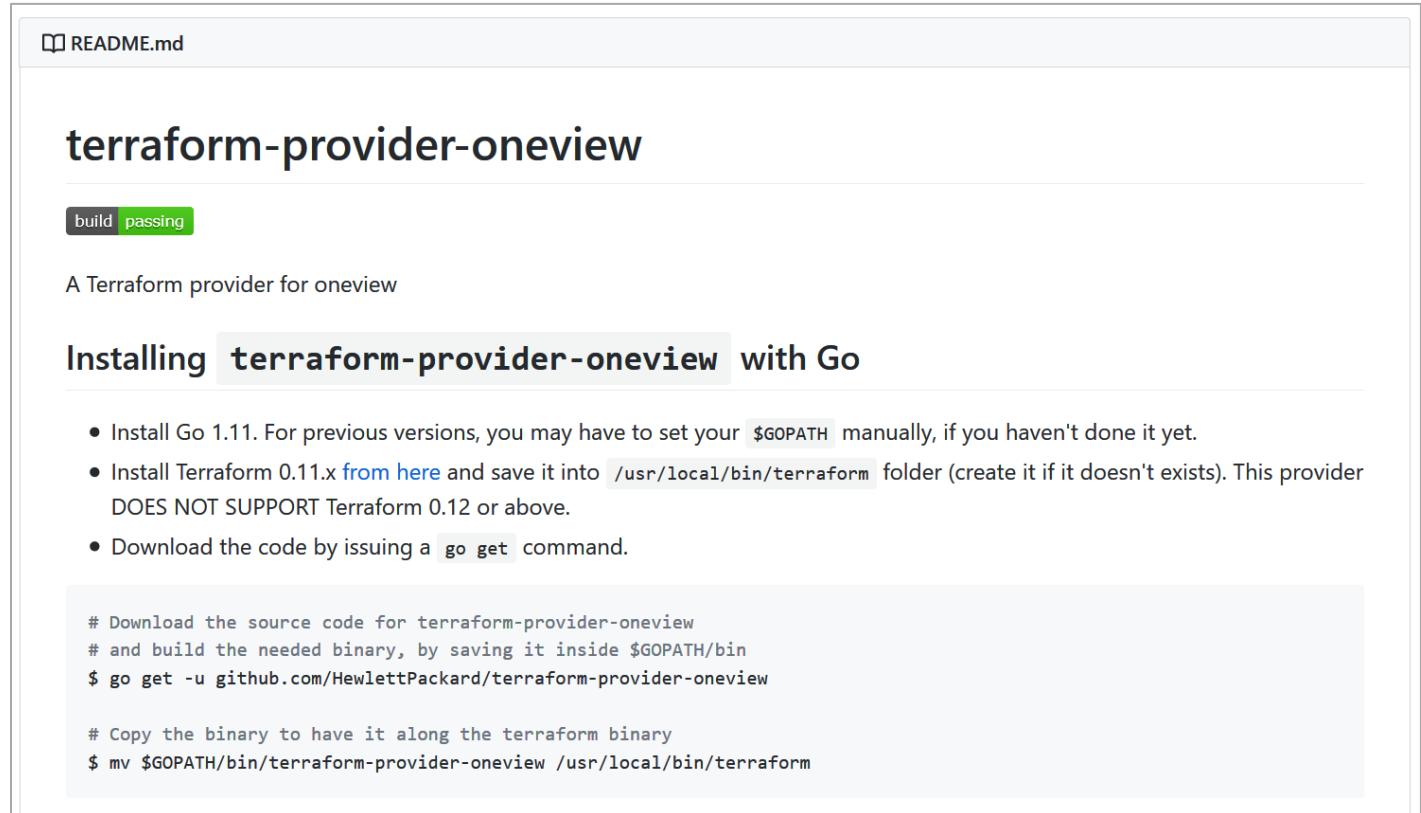
- Then create a **oneview_config.json** file with our appliance information:

```
cat > oneview_config.json << "EOF"
{
    "ip": "192.168.56.101",
    "api_version": 1600,
    "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).

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:



The screenshot shows the GitHub repository for `terraform-provider-oneview`. The `README.md` file is open, displaying the project's purpose: "A Terraform provider for oneview". A green "build passing" badge indicates the build status. Below the README, there is a section titled "Installing `terraform-provider-oneview` with Go" with the following instructions:

- 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:~$
```

DRAFT



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:

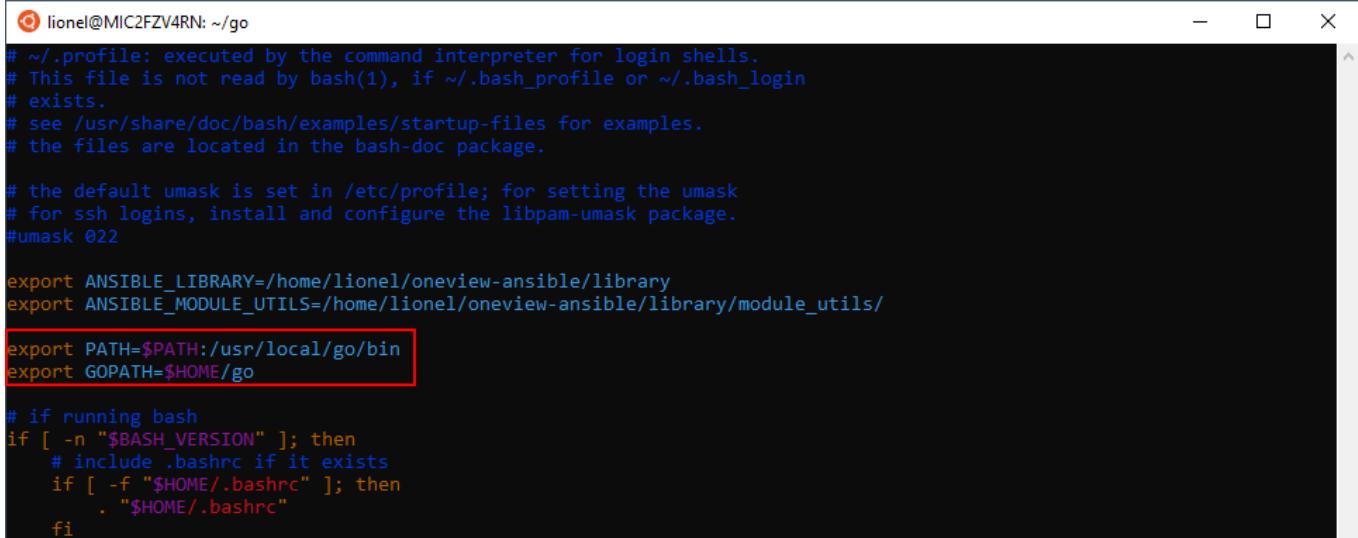
```
wget https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz  
sudo tar -C /usr/local -xzf go1.14.2.linux-amd64.tar.gz
```

- 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



```
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

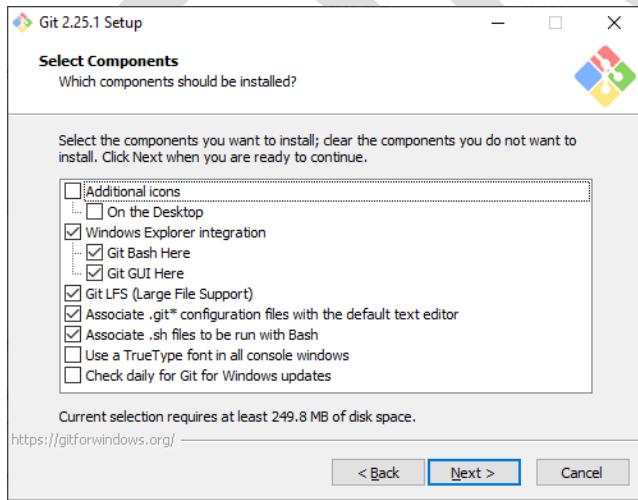
Visual Studio Code does come with an integrated Git source control provider. However, for that to work, Git itself needs to be installed on your Windows system as well.

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.

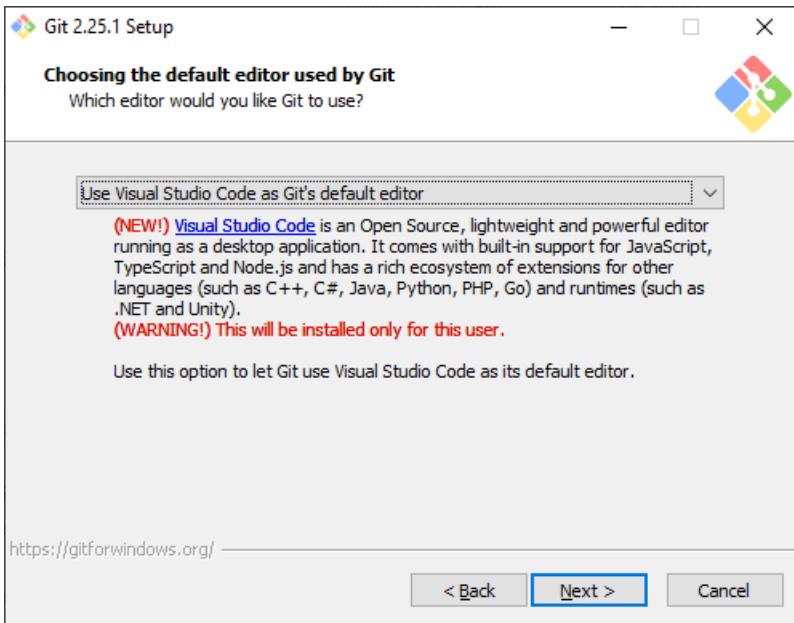
- Go to <https://git-scm.com/> and download and install Git on your machine.
- Select **Windows** and launch the installation:



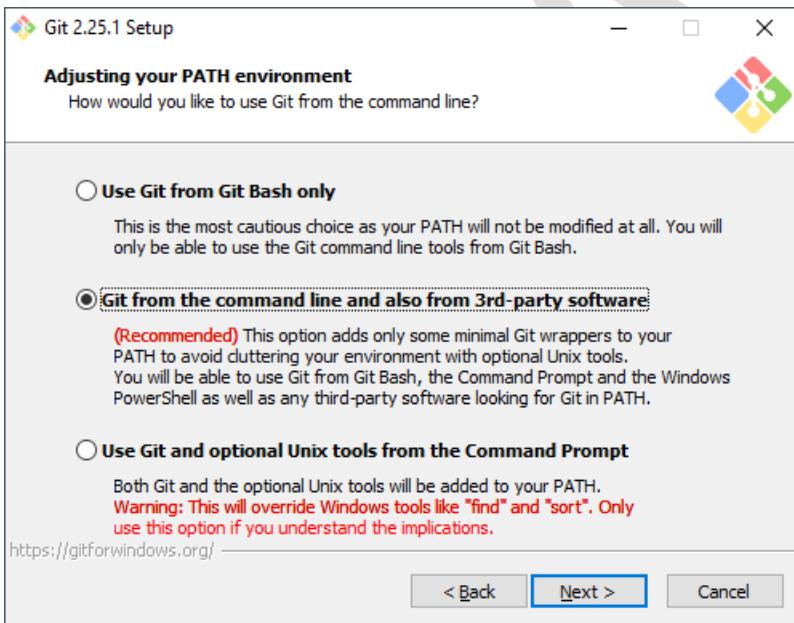
- Keep the proposed default components selected



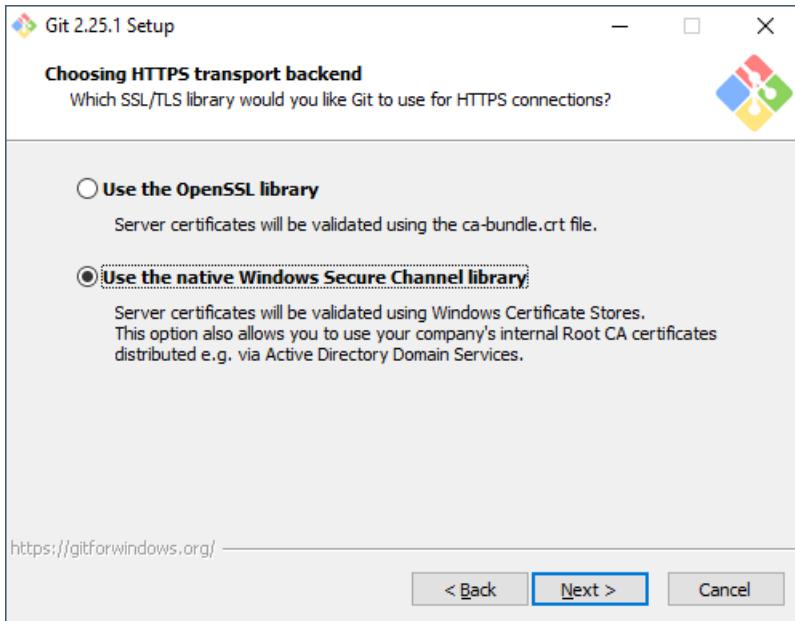
- Choose Visual Studio Code as Git's default editor option



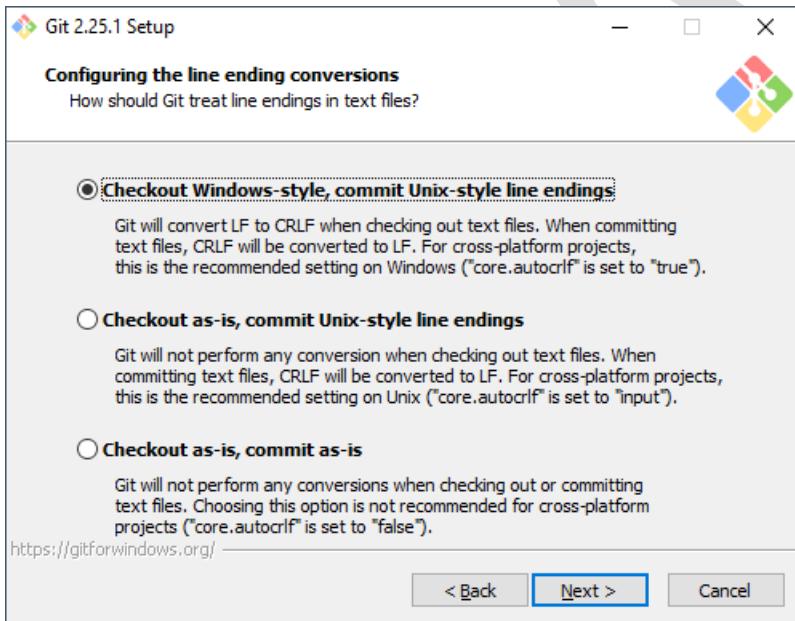
- Keep the recommended PATH environment option selected



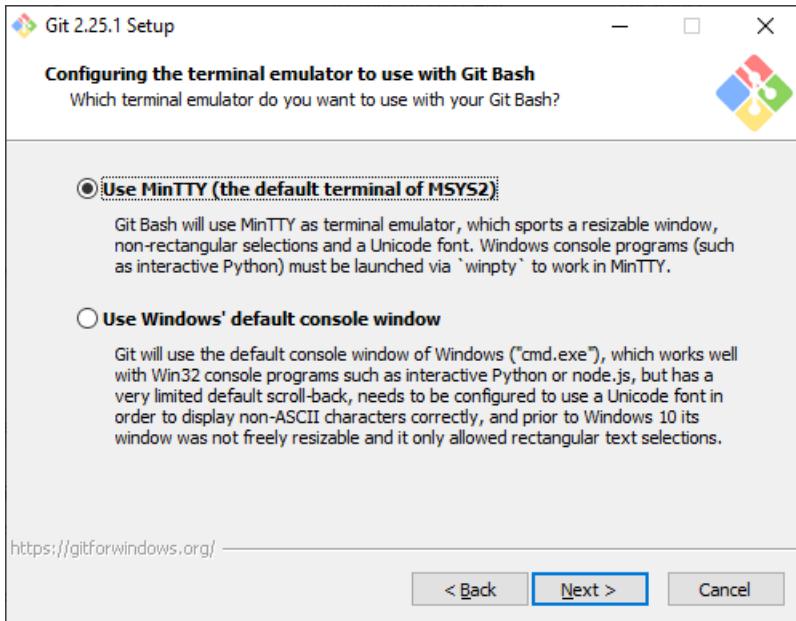
- Use the proper HTTPS transport backend option according to your environment. For HPE employees, I would recommend using the native **Windows Secure Channel library**:



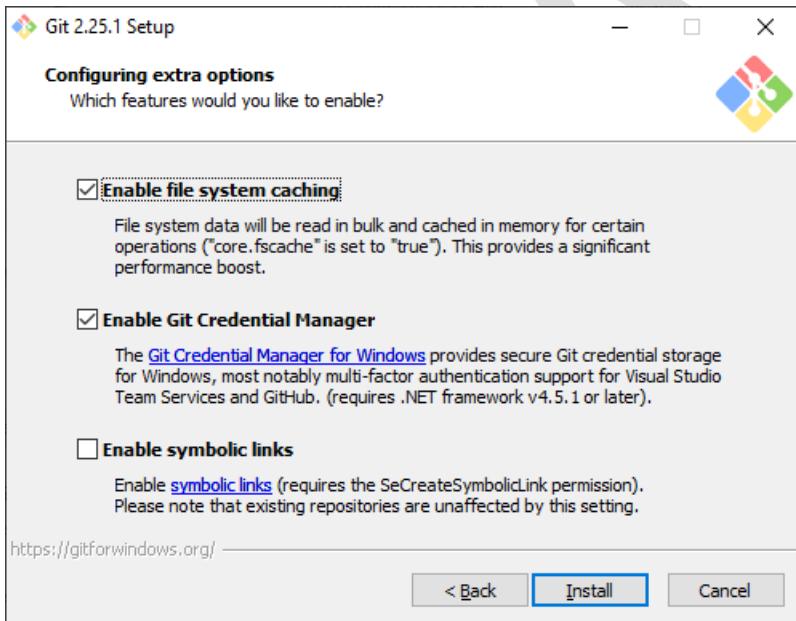
- Use the **Checkout Windows-style** as it is the recommended setting on Windows:



- 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.

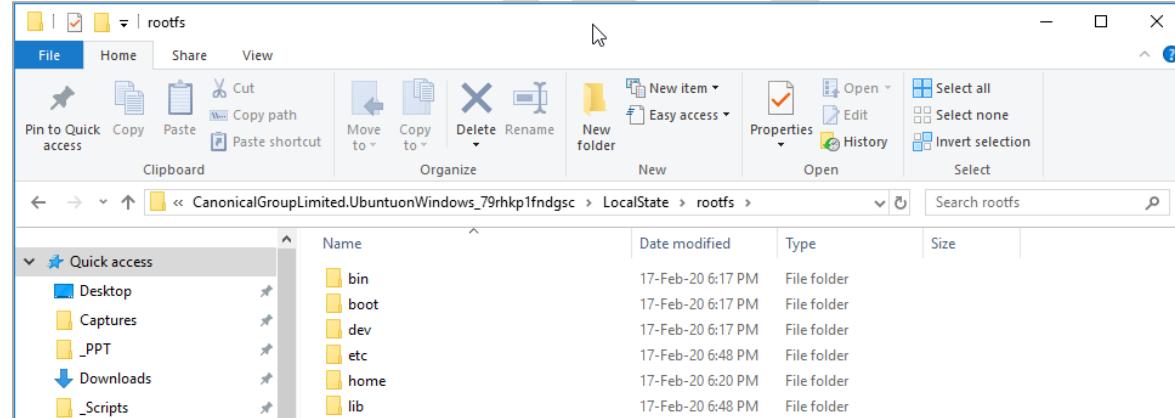
/mnt/c/ shortcut maps to c:\ Windows drive

```
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.

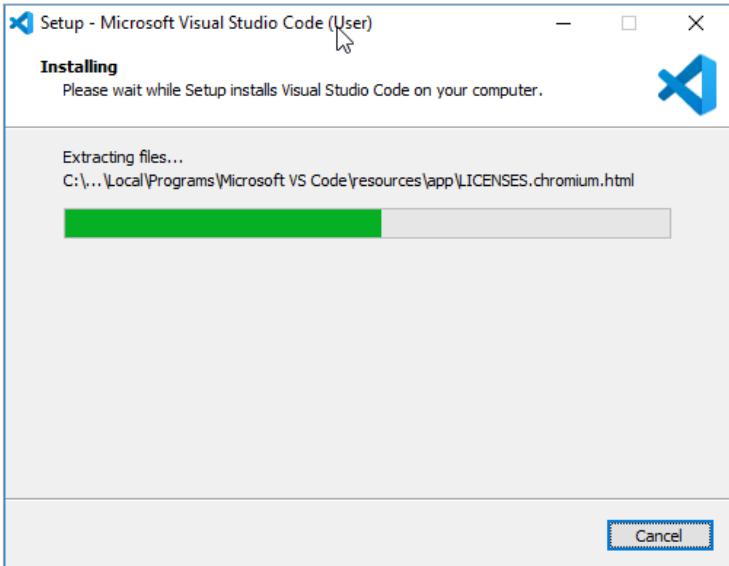
Download Visual Studio Code

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 "User Installer" link (highlighted in yellow) and a "System Installer" link. Below the Linux section, there are links for ".deb", ".rpm", and ".tar.gz" files, each with "64 bit" and "32 bit" options. A "Snap Store" link is also present. The page has a clean, modern design with the Microsoft logo at the top left.

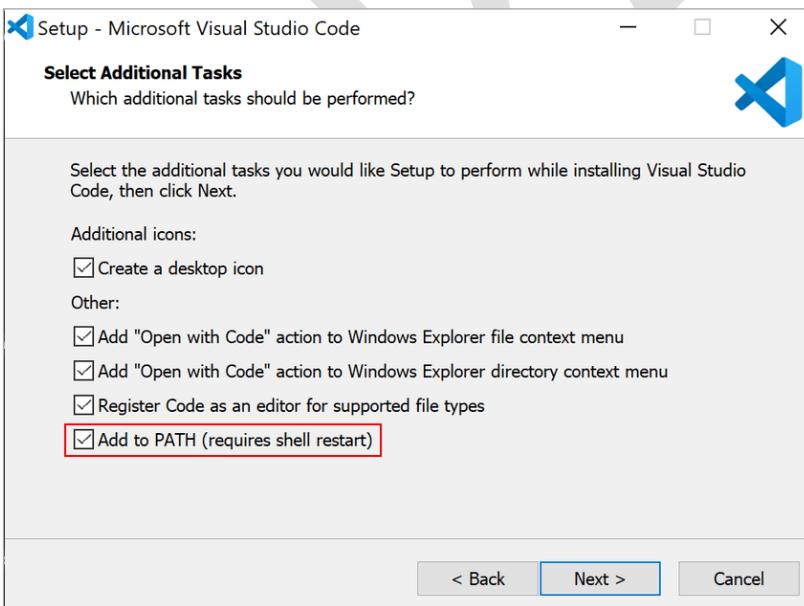


- You can proceed with the installation using all default parameters.



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

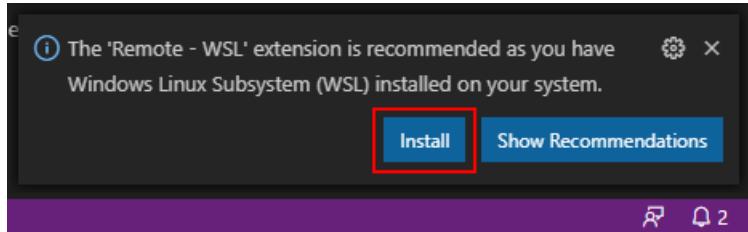
- When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily interact with WSL.



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

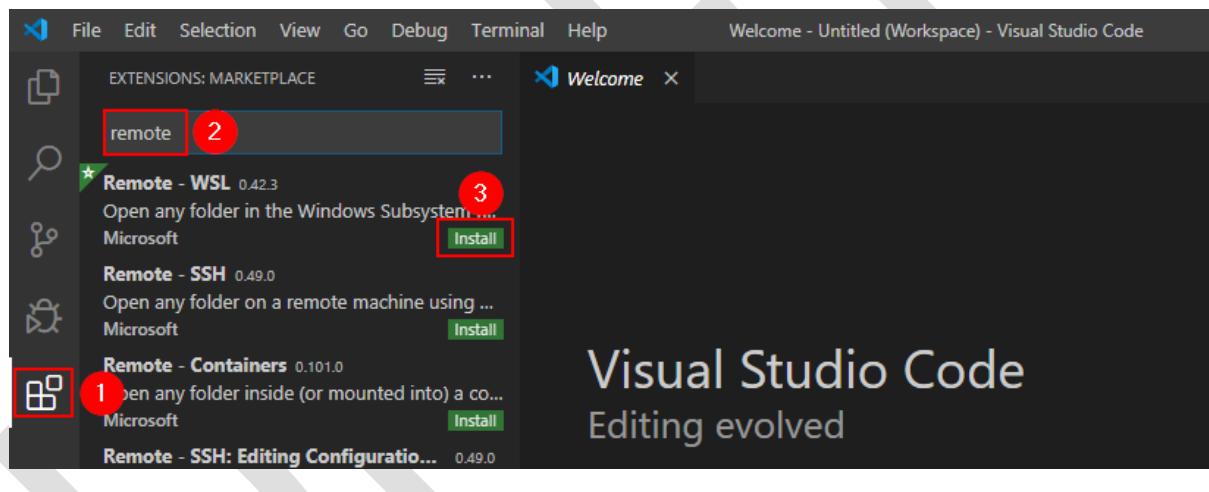
When VS Code is opening, you should notice a warning message saying "The 'Remote - WSL' extension is recommended as you have Windows Linux Subsystem (WSL) installed on your system"

- Click on **Install**

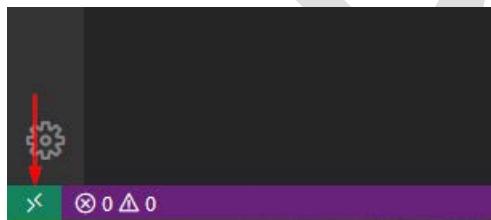


This is the extension that VS Code need to connect to our Ubuntu running in the Windows Subsystem for Linux.

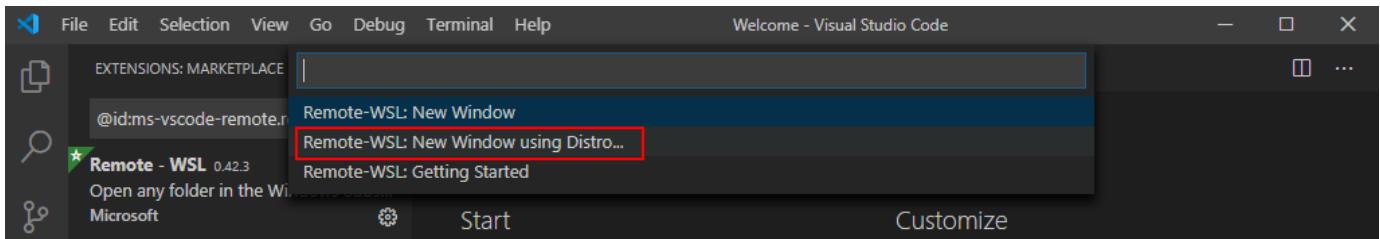
Note: If the message is not appearing, you can install manually this extension by clicking on the **Extensions** icon on the Activity bar and search for **Remote - WSL**



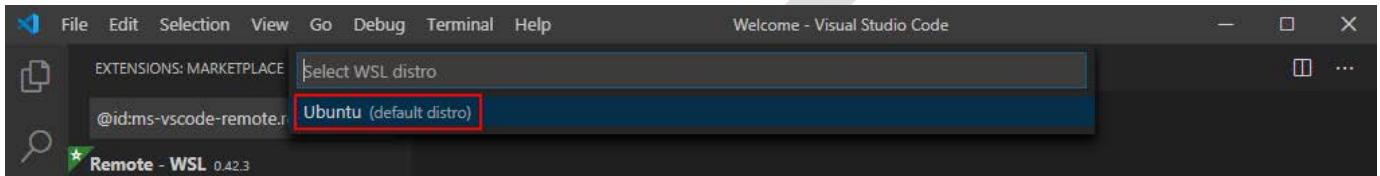
- Once installation complete, click on the Remote "Quick Access" status bar item in the lower left corner to get a list of the most common commands.



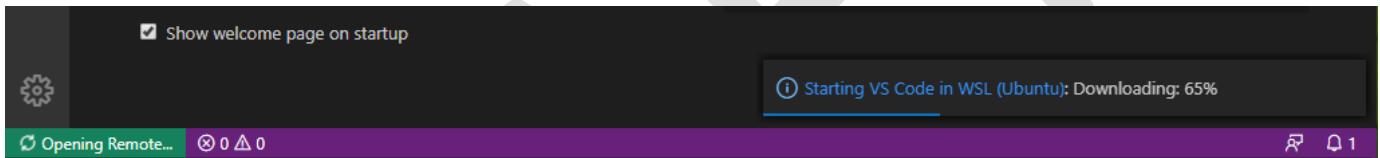
- Select **Remote-WSL: New Window using Distro...**



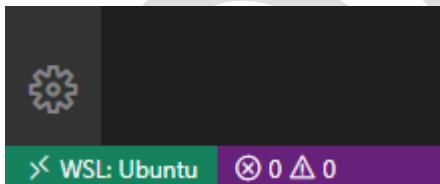
- Then select **Ubuntu**



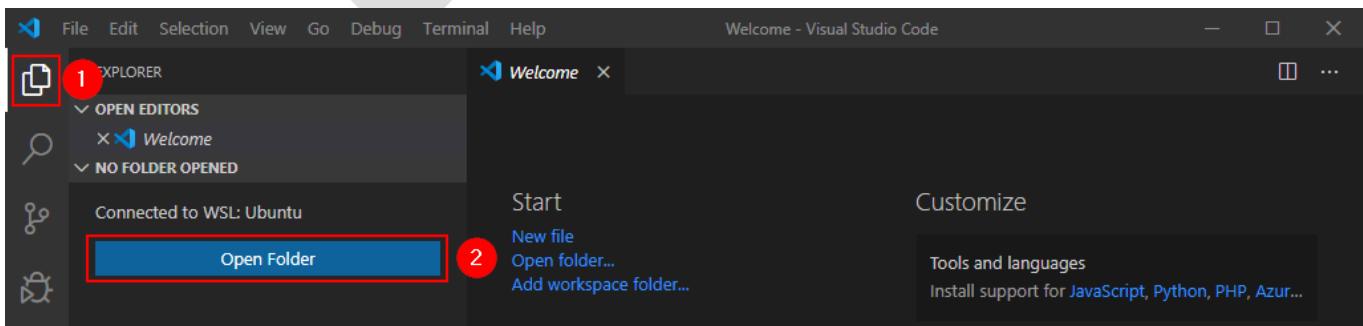
- A Remote connection to Ubuntu is taking place in a new VS Code window:



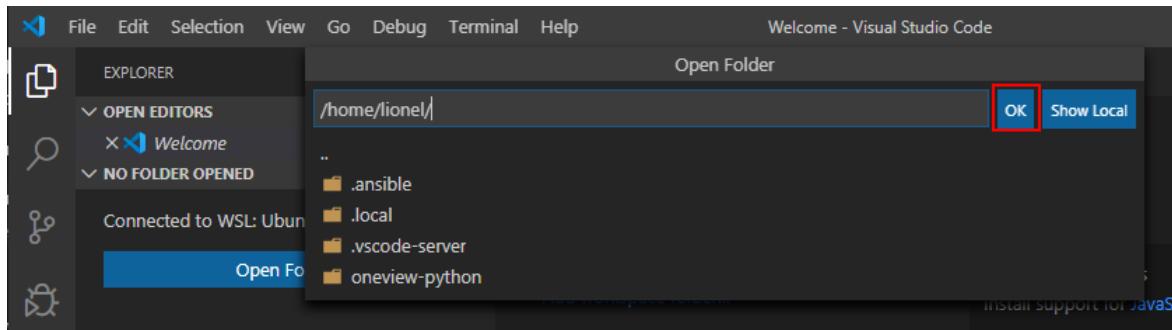
- You can close the previous VS Code window.
- Once connected, in the lower left corner of the Status Bar, you should see that you're connected to your WSL: Ubuntu instance.



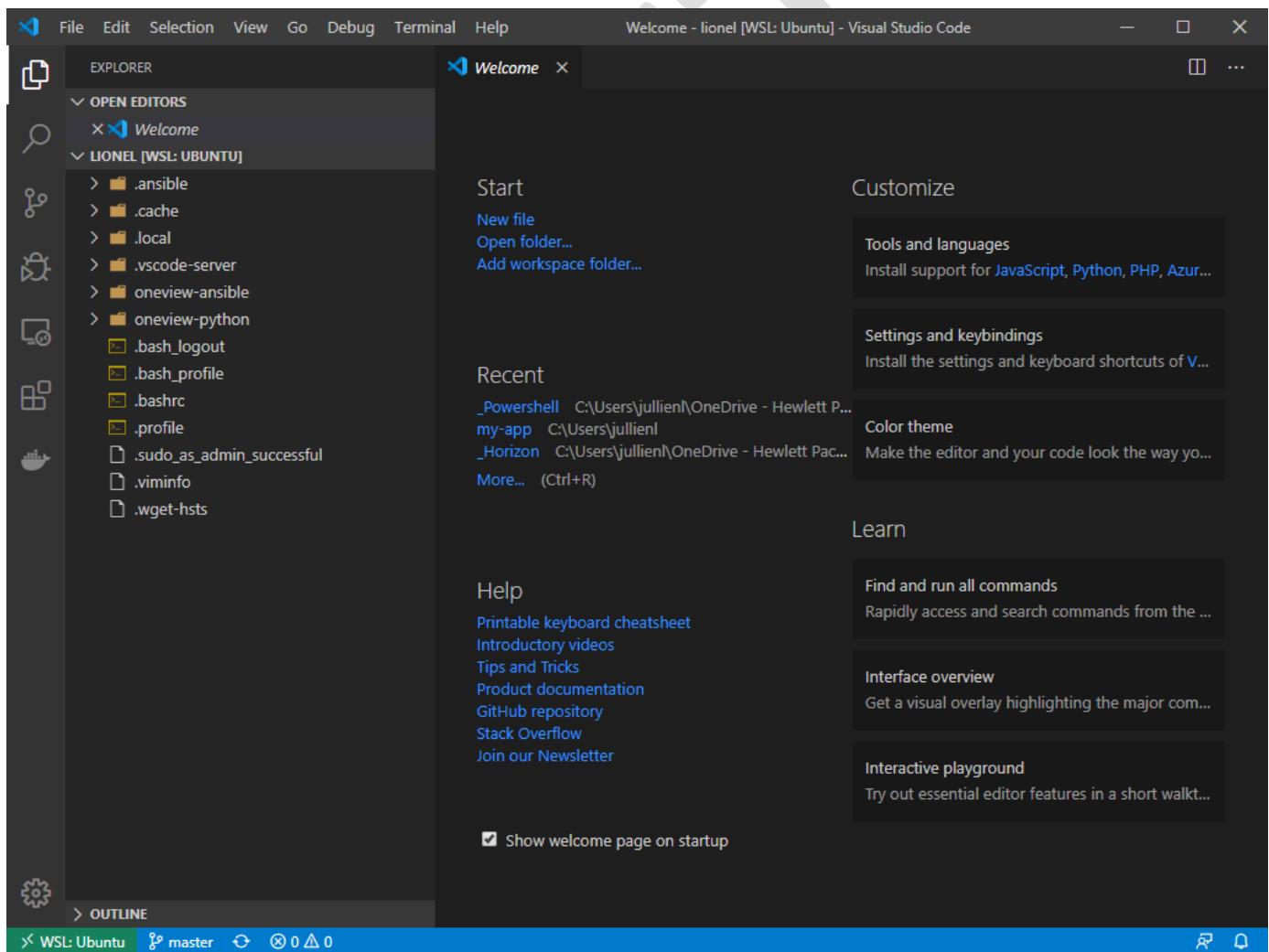
- You can then open a folder located on the WSL:Ubuntu using the **Explorer** icon on the Activity bar



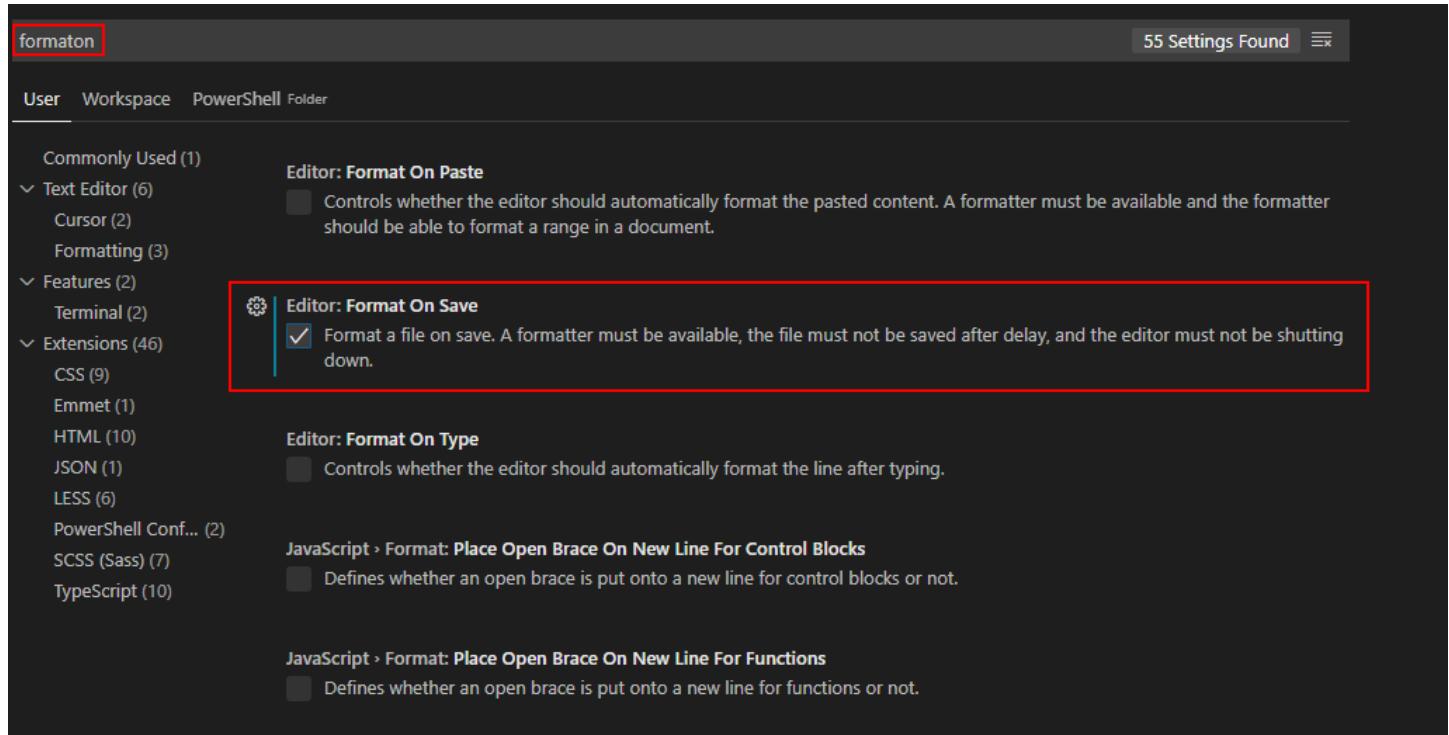
- Click **OK** to the `/home/<username>` folder



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

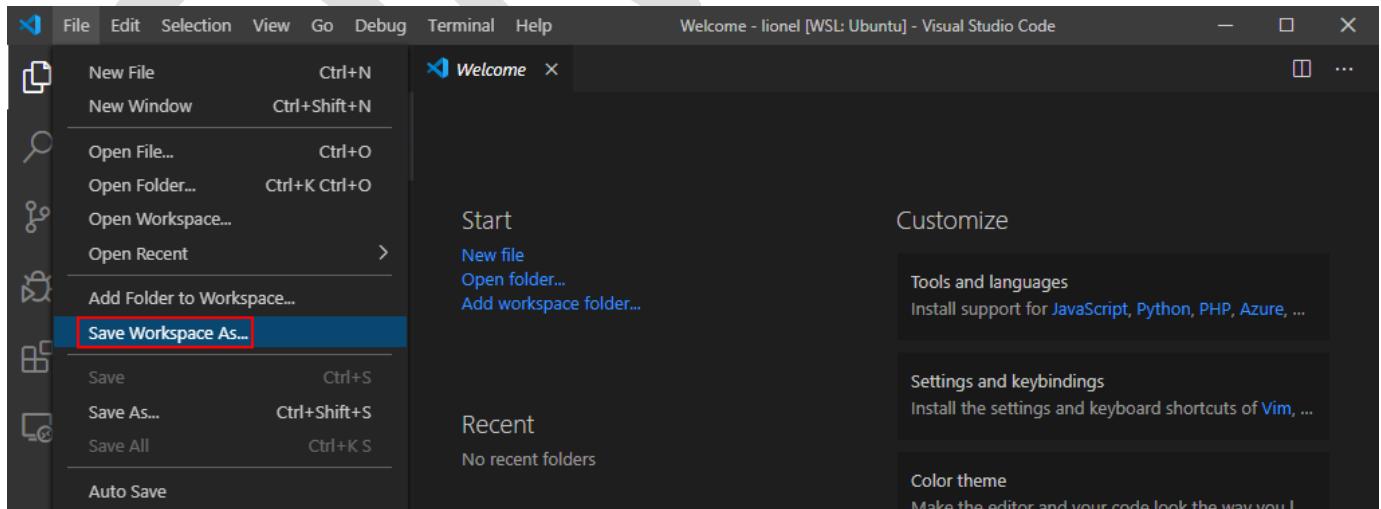


There are tons of VS Code settings that you can tweak but there is one that is really recommended when writing scripts is the **Format On Save**. This “must have” option automatically formats your code when you save your work by improving spacings, indents, by wrapping lines if necessary and fixing quotes.

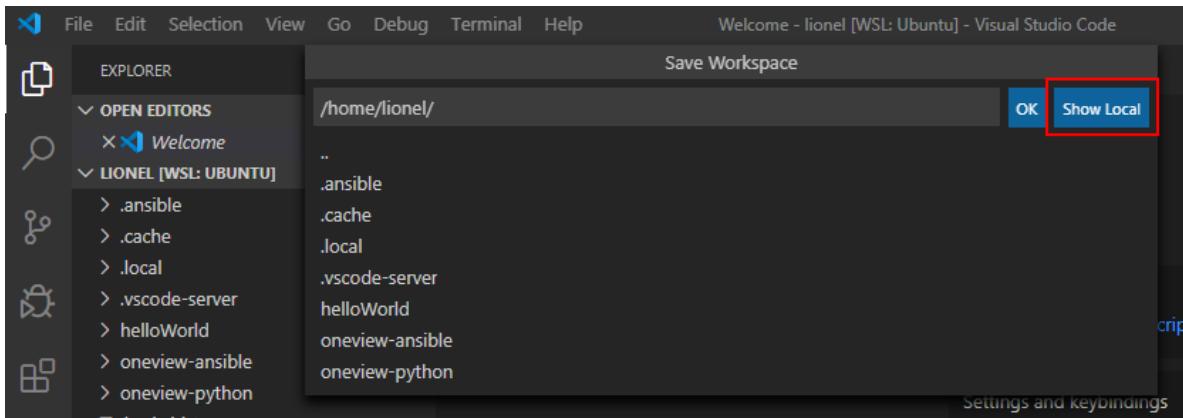


The next step is to save the current VS Code configuration in a workspace. A VS Code "workspace" is usually just your project root folder.

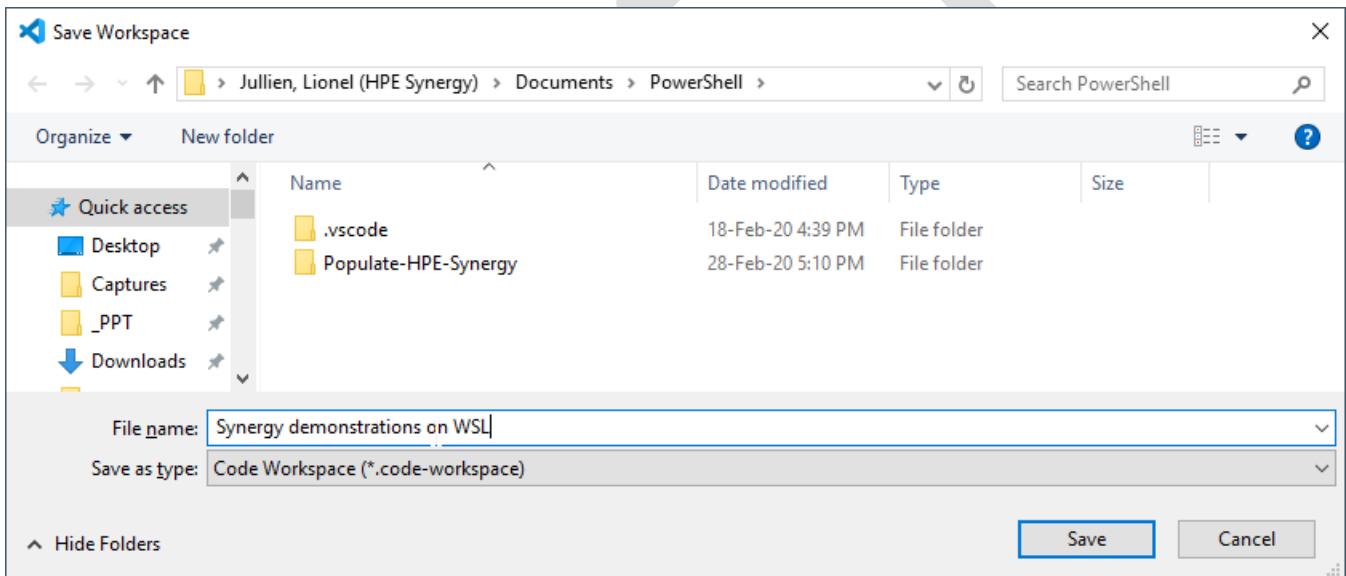
- Select **File** menu and click on **Save Workspace as...**



- Select **Show Local** and select a folder in your Windows Documents folder



- Select an emplacement 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 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.

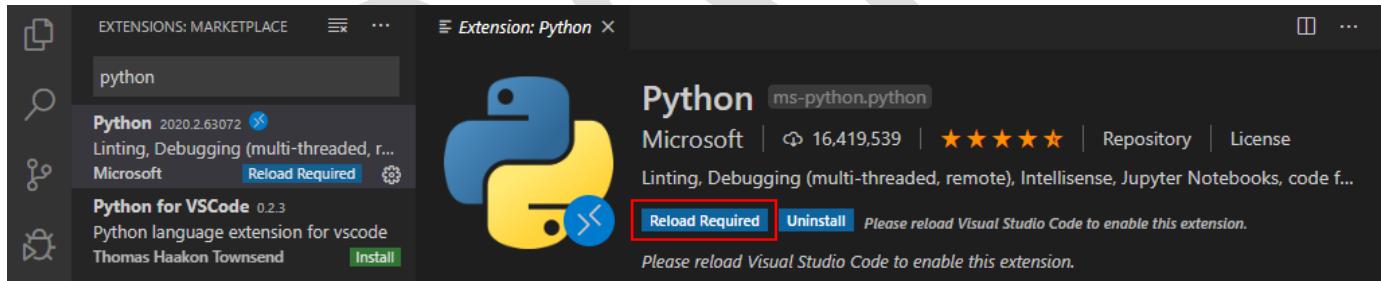
There are popular ones for PowerShell, Python, Ansible...

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



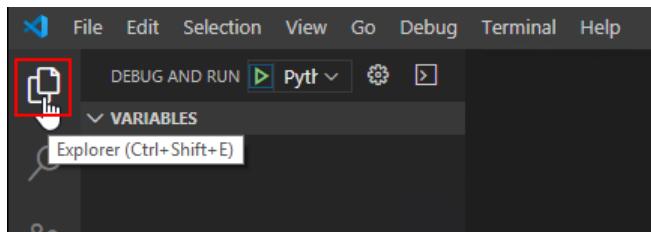
This extension provides nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to module documentation, ability to run the active script file in the VS Code terminal console, etc.

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

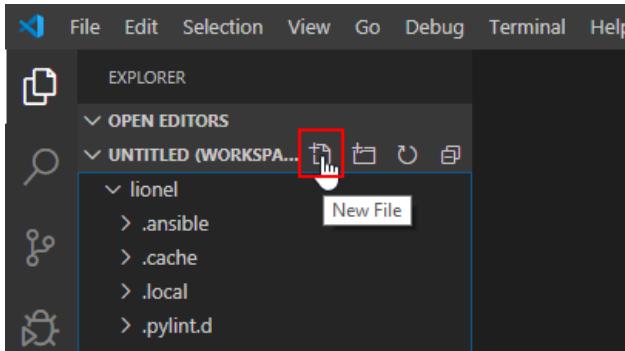


Note: For the Python extension, we are going to use the Python interpreter running in our Ubuntu Linux distribution hosted by WSL. Another good option if you don't want to use WSL would be to run Python directly on your Windows system, see <https://www.python.org/downloads/windows/> and <https://docs.microsoft.com/en-us/windows/python/beginners>

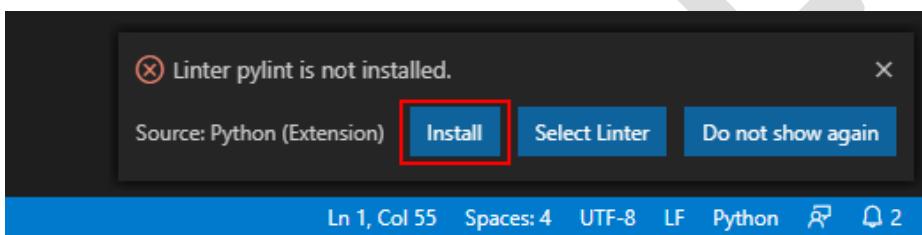
- Let us now test the Python interpreter, click now on **Explorer** on the Activity Bar



- Then click on **New File**



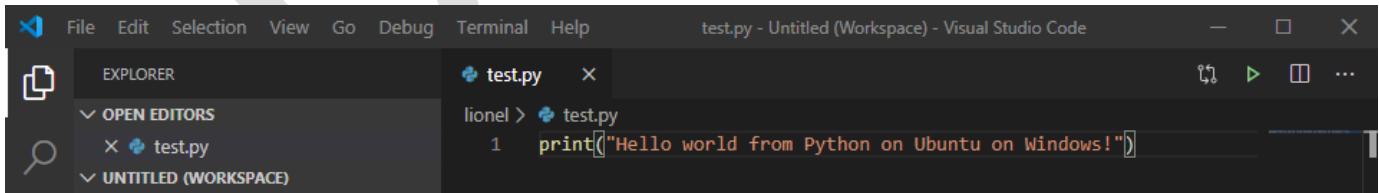
- Enter a name like **test.py** then press **ENTER**
- You can install pylint by clicking on **Install** when the window pops-up at the creation of the Python script. A code linter is a program that analyses your source code for potential errors and code style guideline violations. Pylint is one of the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

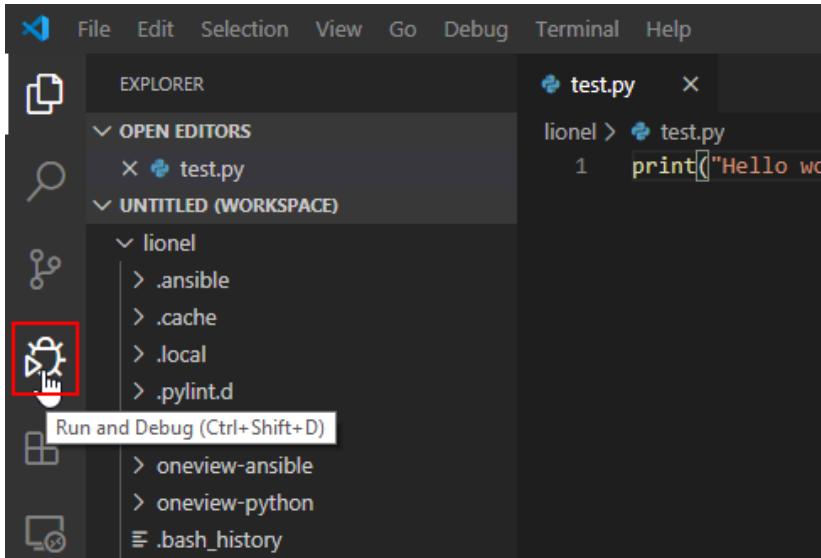
- Enter the following line to build the Python script test.

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

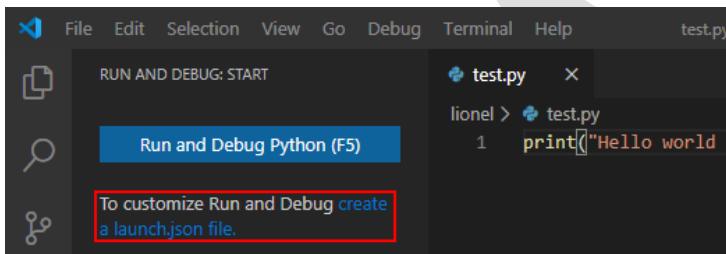


- Save the file by pressing **CTRL + S**

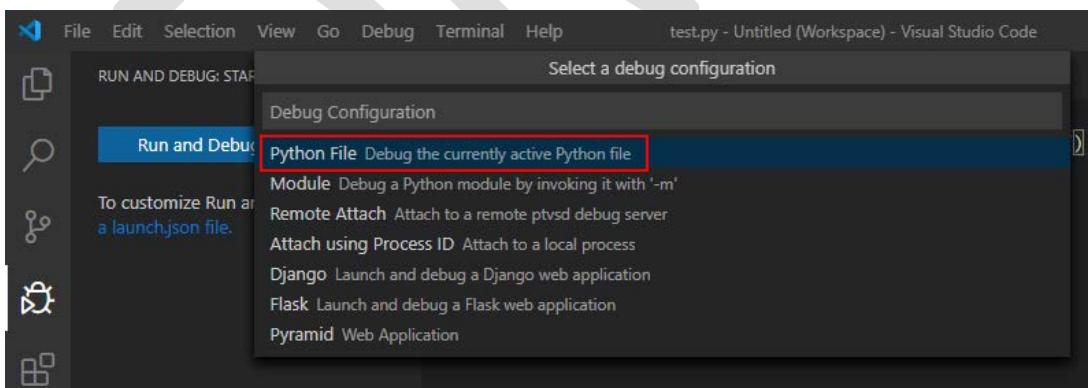
- Click on **Run and Debug**



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



- Select **Python File**

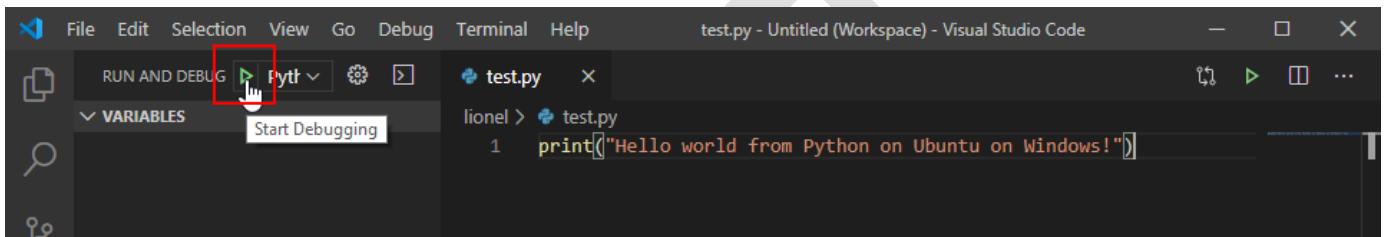


- To complete the Python debug and Run configuration, press **CTRL + S** to save the configuration file then **CTRL + F4** to close the configuration file.

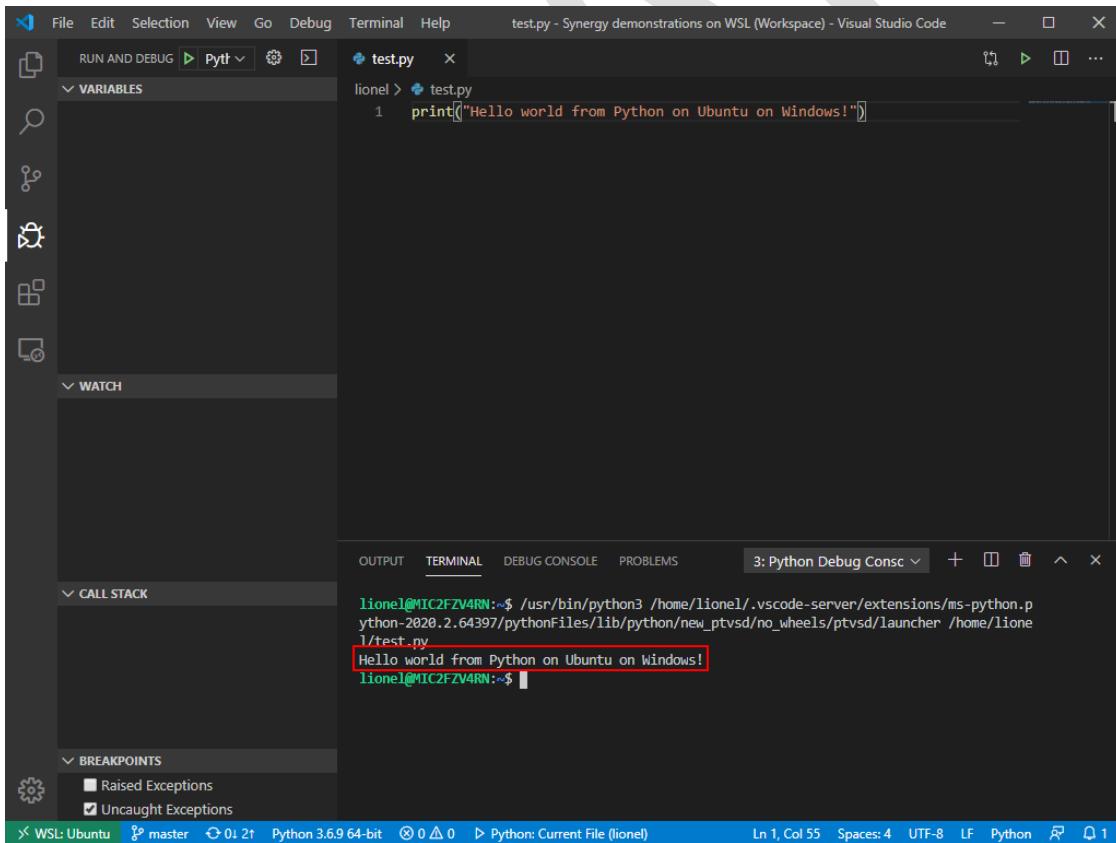
Tip: You can see the Python interpreter version in use in the status bar



- Now click on **Start Debugging**



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

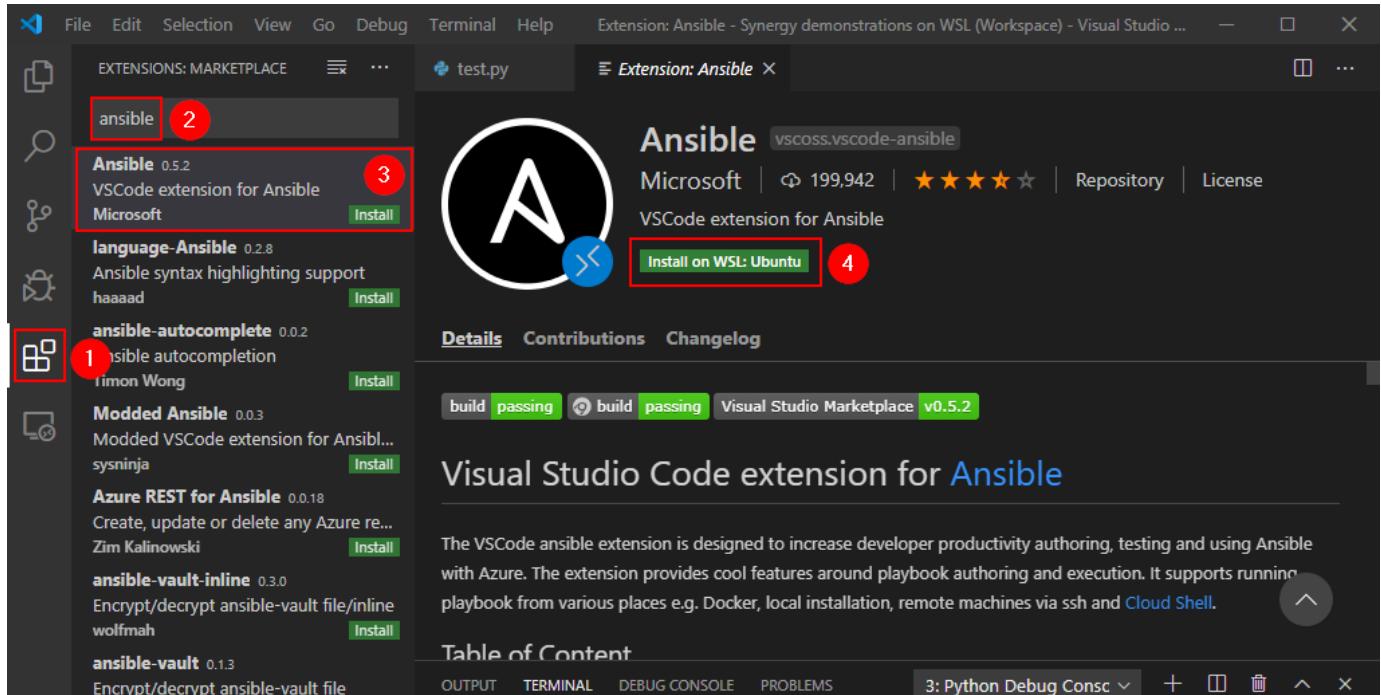


We are now all set for Python scripting and debugging.

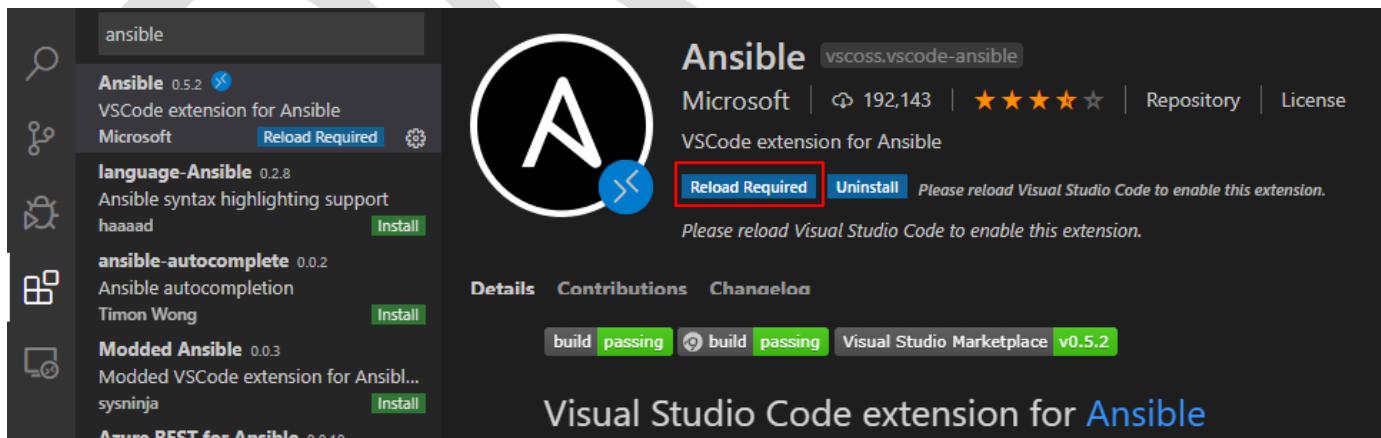
Preparing VS Code for Ansible

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.



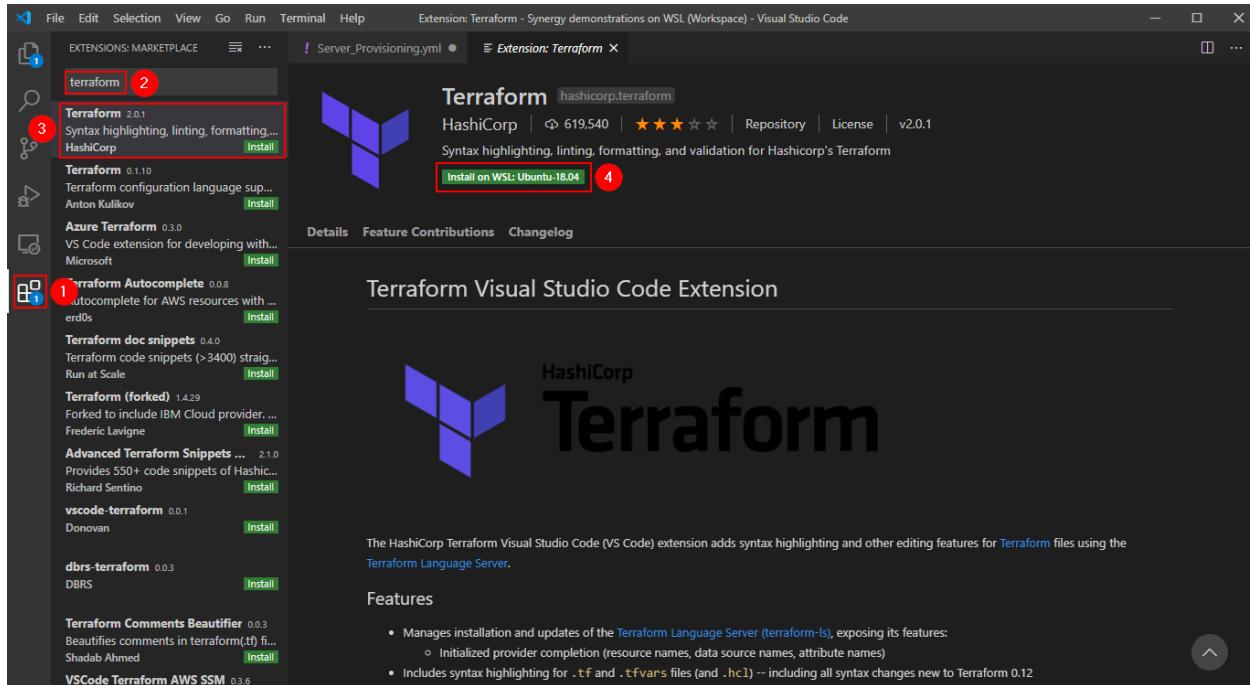
- Once the installation is complete, click on **Reload Required** to activate the extension.



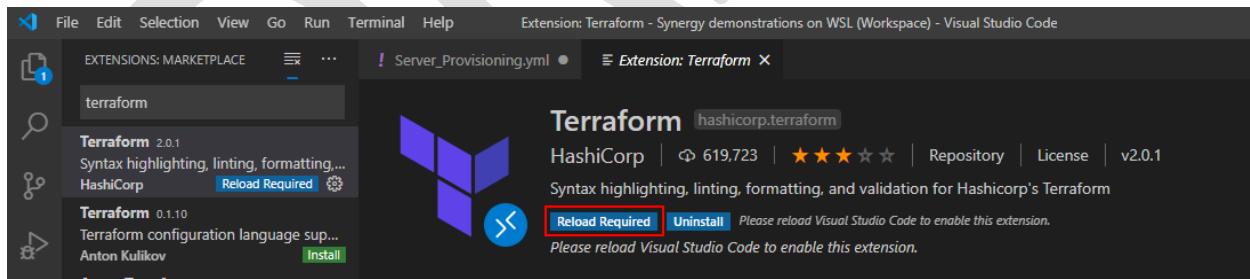
Preparing VS Code for Terraform

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.



Preparing VS Code for PowerShell

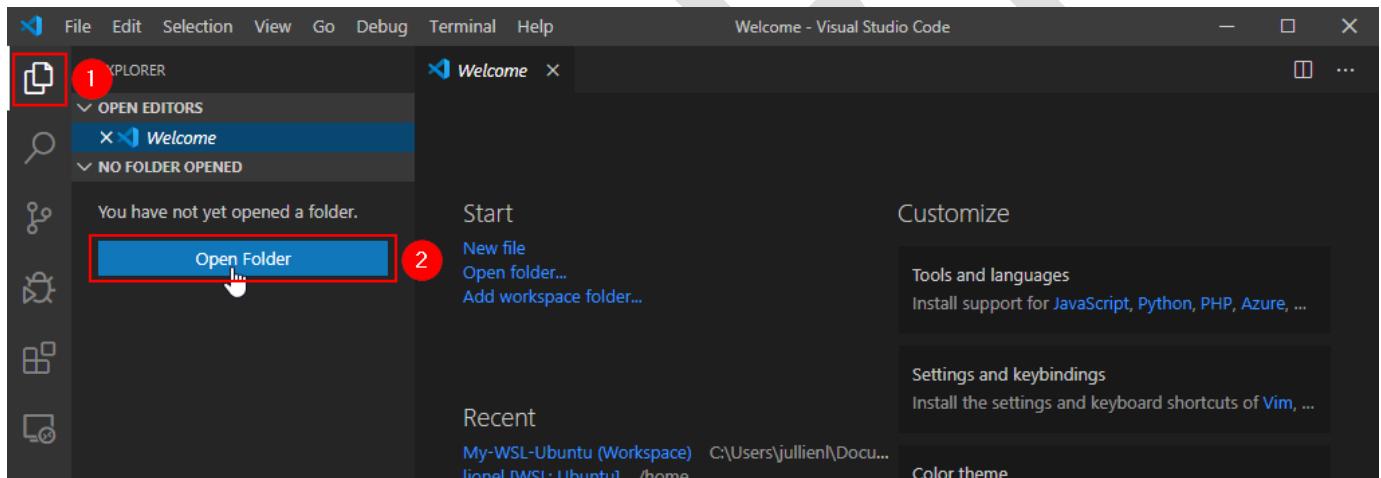
Next, we can install the PowerShell extension to get also nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to cmdlet documentation, ability to run the active script file in the VS Code terminal console, etc.

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:

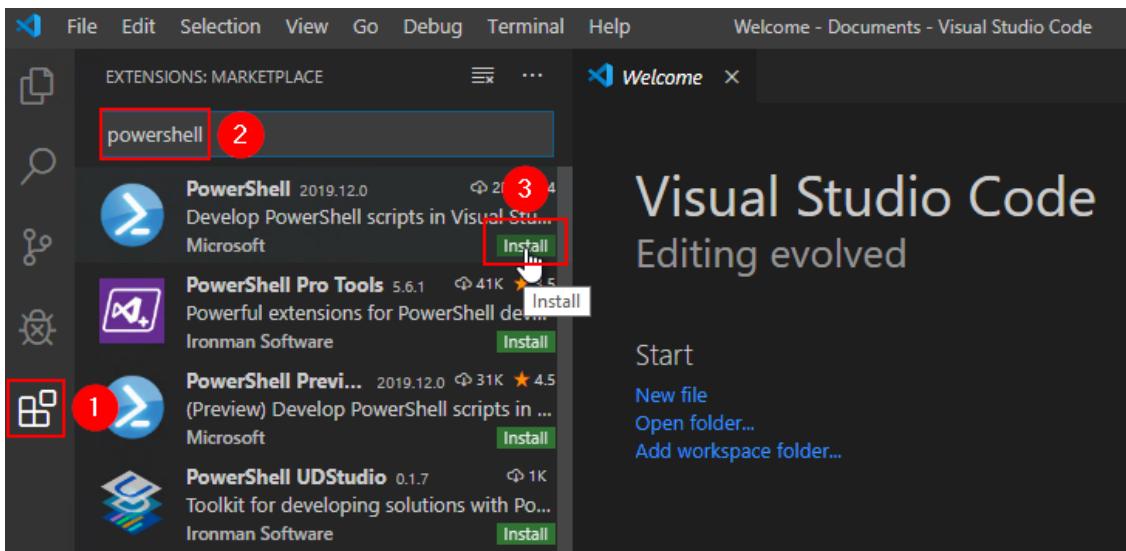


- Then click on the **Explorer** icon on the Activity bar and then click **Open Folder**.

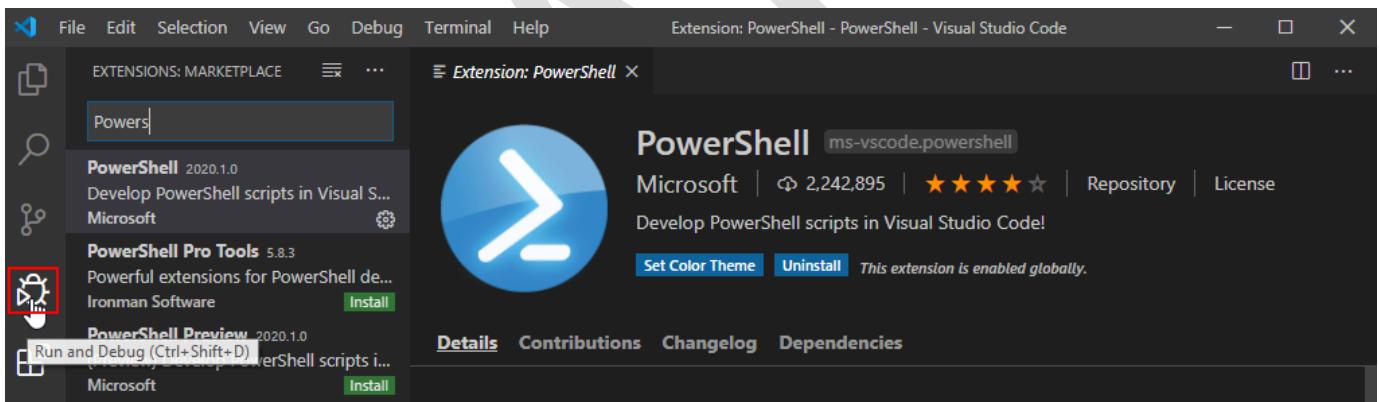


- Open the folder you want your PowerShell project to be in (e.g. \Documents\PowerShell) and click **Select Folder**.

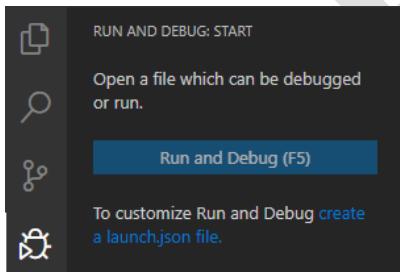
- Go to the **Extensions** pane and search for the **PowerShell** extension and click on the **Install** button:



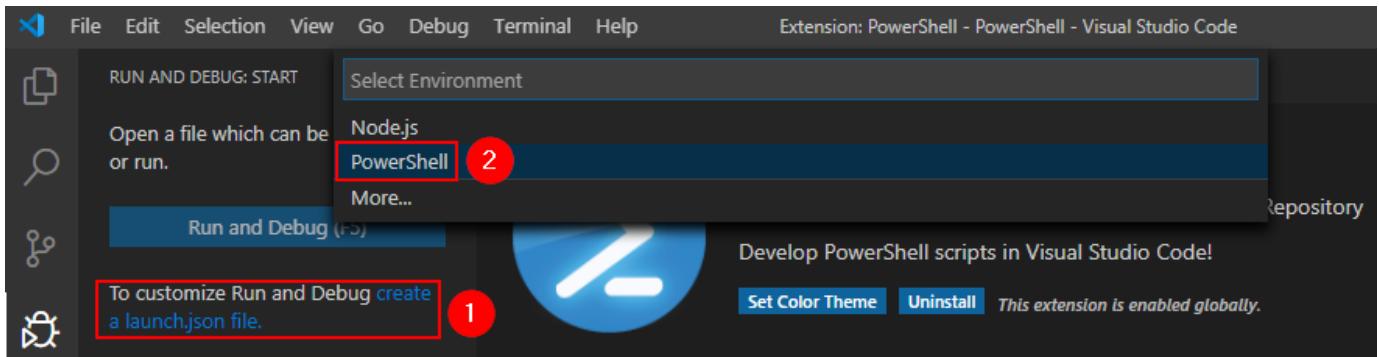
- To configure the PowerShell interpreter once the extension is installed, click on **Run and Debug** on the Activity Bar



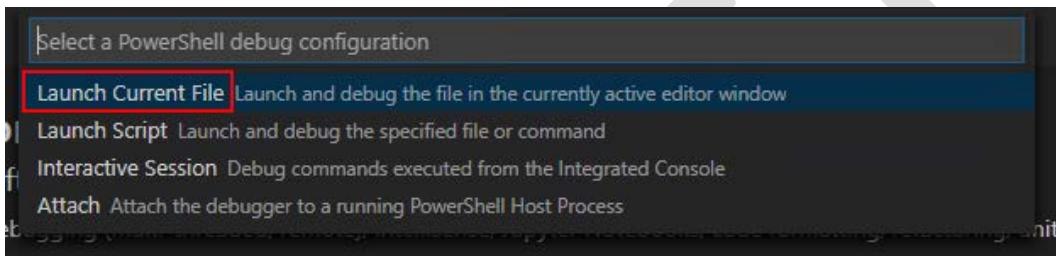
Visual Studio Code's built-in debugger needs some quick setup to run and debug PowerShell scripts. If your system is already configured, you can skip this section. When VS Code is not configured for PowerShell, the following dialog is displayed:



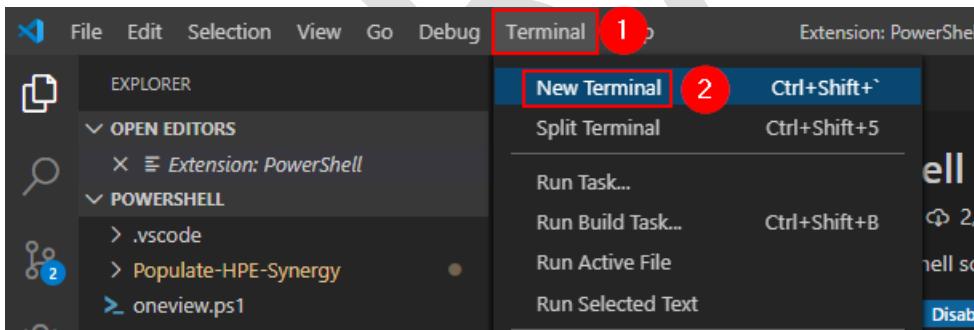
- Click on **Create a launch.json file** to configure Run and Debug, then select **PowerShell**



- Then select **Launch Current File**



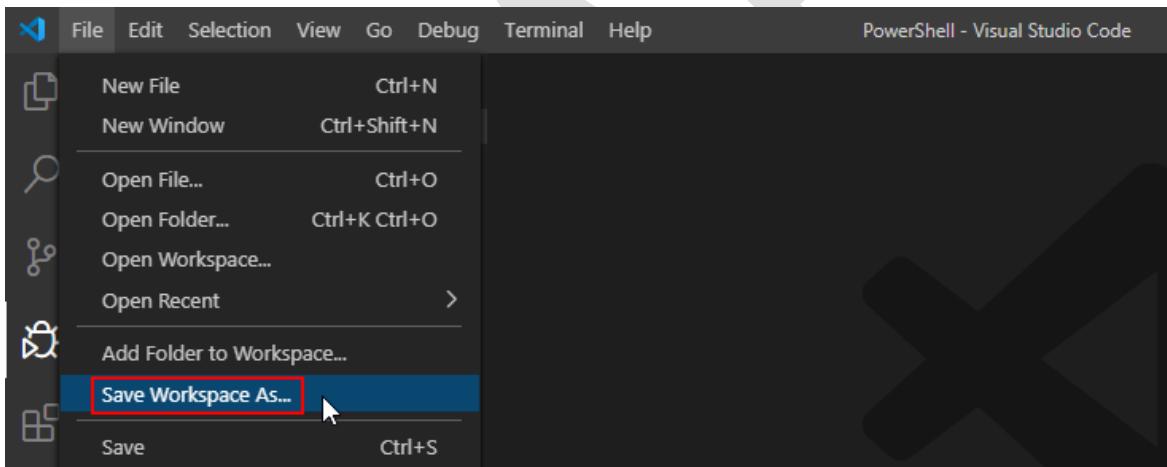
- Then press **CTRL + S** to save the configuration file then **CTRL + F4** to close the configuration file.
- Select now **Explorer** in the activity bar, a PowerShell Integrated console should start, or you can select **Terminal / New Terminal**



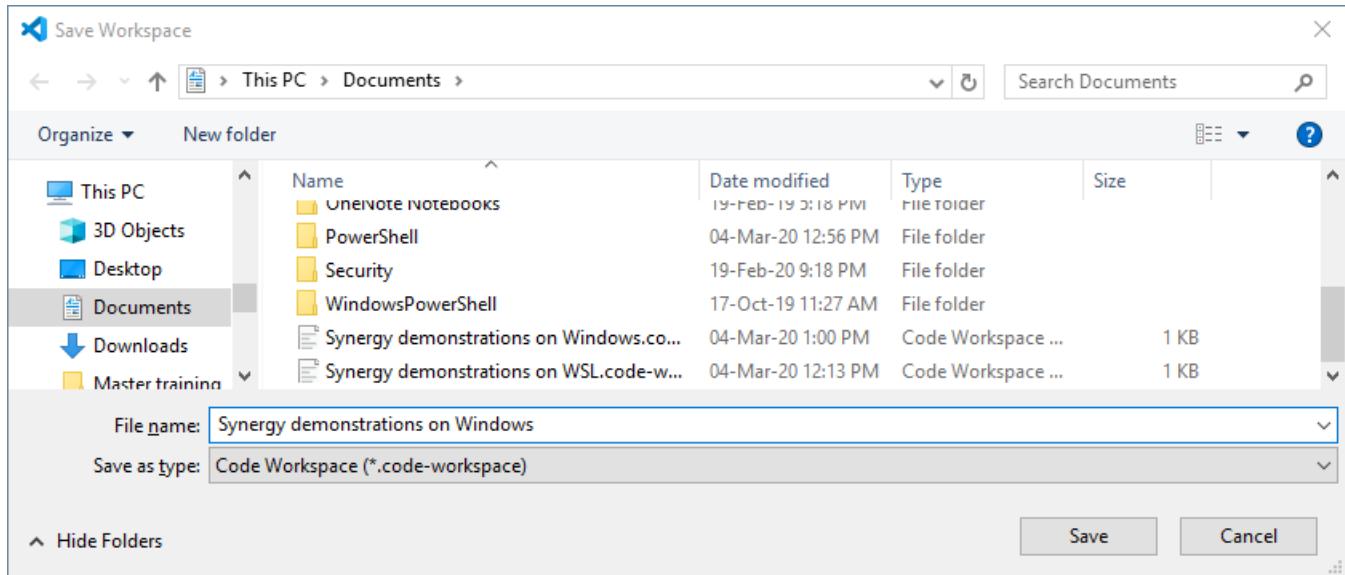
- You can enter **get-service** to make a quick test:

Last step is to save the workspace as our PowerShell workspace project.

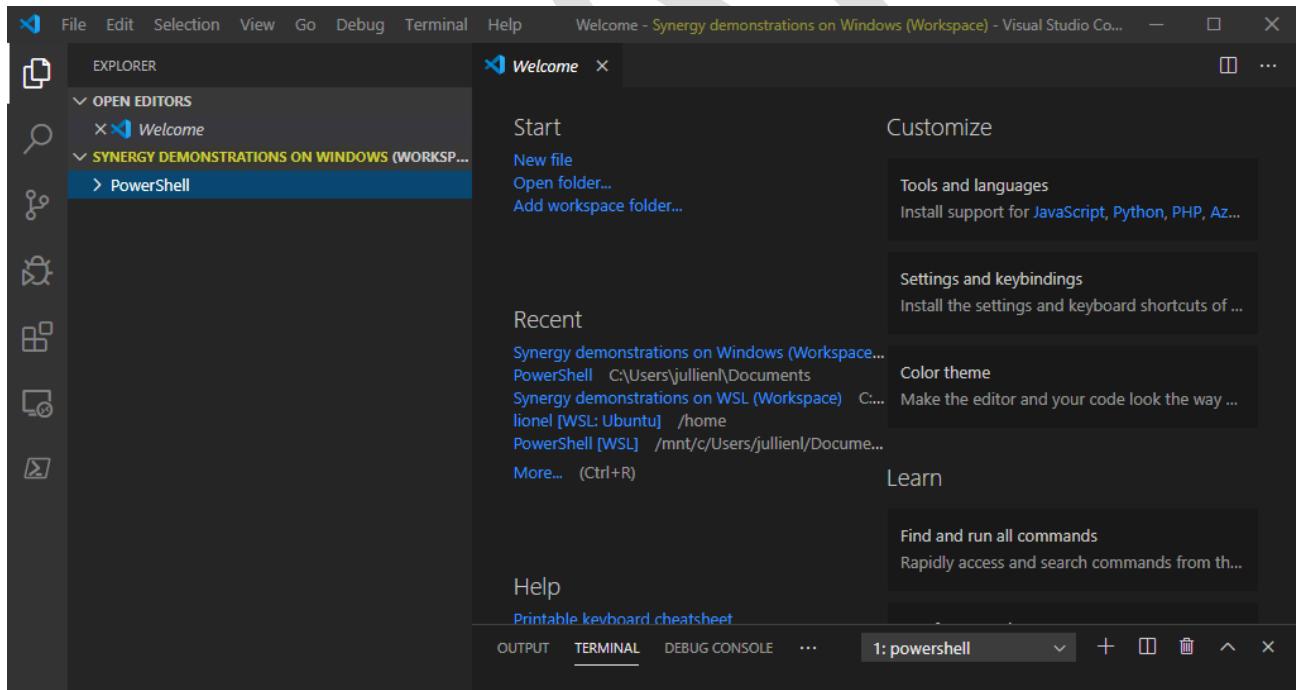
- Select the **File** menu then select **Save Workspace As...**



- Select your Windows User Home directory, enter a project 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.

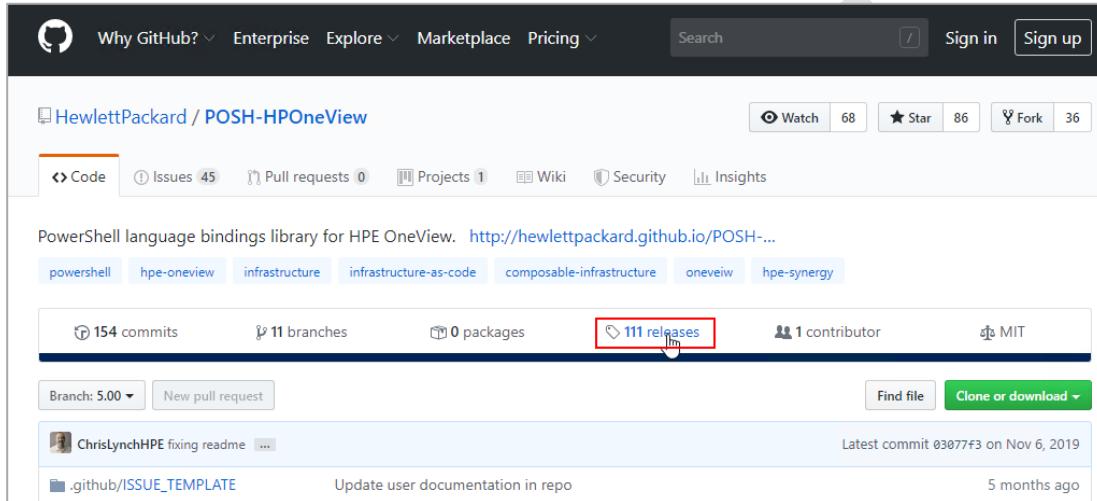


- You can also set the **Format on Save** option in **Preferences** to automatically format your code on save.

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 HPOneView.520
```

A screenshot of the Visual Studio Code interface. The top navigation bar includes 'DEBUG CONSOLE', 'PROBLEMS', 'OUTPUT', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the title '2: PowerShell Integrated'. The terminal window displays the following text:

```
=====> PowerShell Integrated Console <=====  
  
PS C:\Users\jullienl\OneDrive - Hewlett Packard Enterprise\PPT\Scripts\_Powershell> Install-Module hponeview.520  
  
Untrusted repository  
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "No"): y
```

- 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 HPOneView.520
```

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

```
get-hpovversion
```

```
PS C:\Users\jullienl\OneDrive - Hewlett Packard Enterprise\PPT\Scripts\_Powershell> Get-HPOVersion  
  
LibraryVersion Path  
-----  
5.20.2422.3962 C:\Users\jullienl\OneDrive - Hewlett Packard Enterprise\Documents\WindowsPowerShell\Modules\hponeview.520...
```

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
```

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.

This completes the PowerShell configuration and concludes Chapter-3

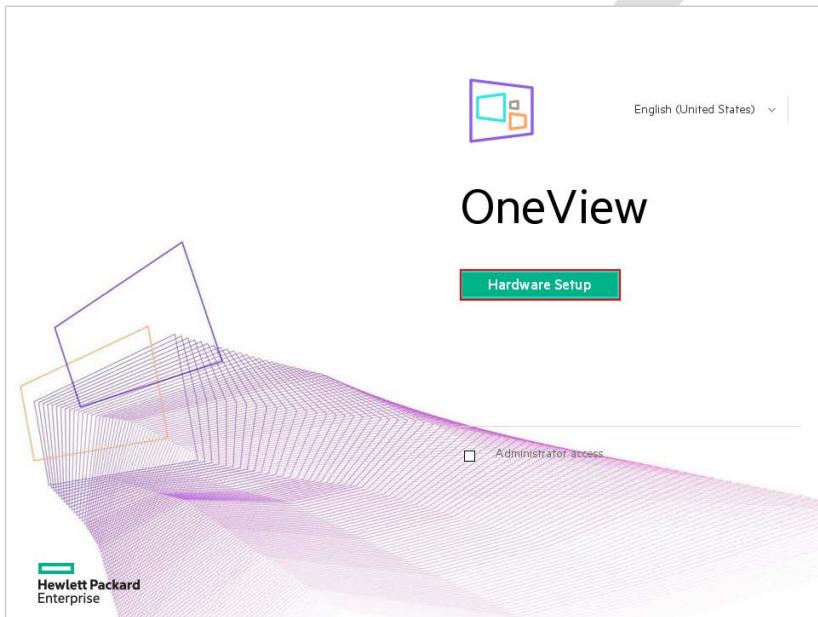
In the next chapter, we will configure HPE OneView.

Chapter-4 -Initial Configuration of the Demonstration Appliance

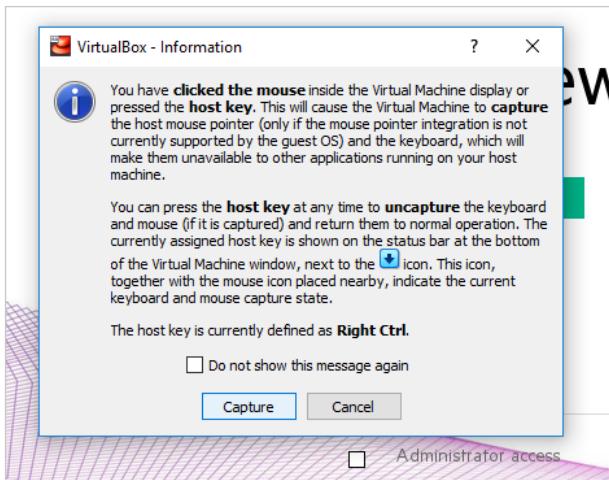
The installation of the scripting languages and the preparation of the Ansible environment is now complete, and OneView has certainly finished booting, we can now finalize the configuration of the appliance.

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

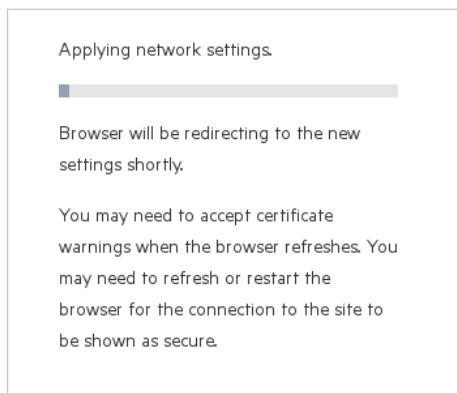
Subnet mask or CIDR

Gateway address

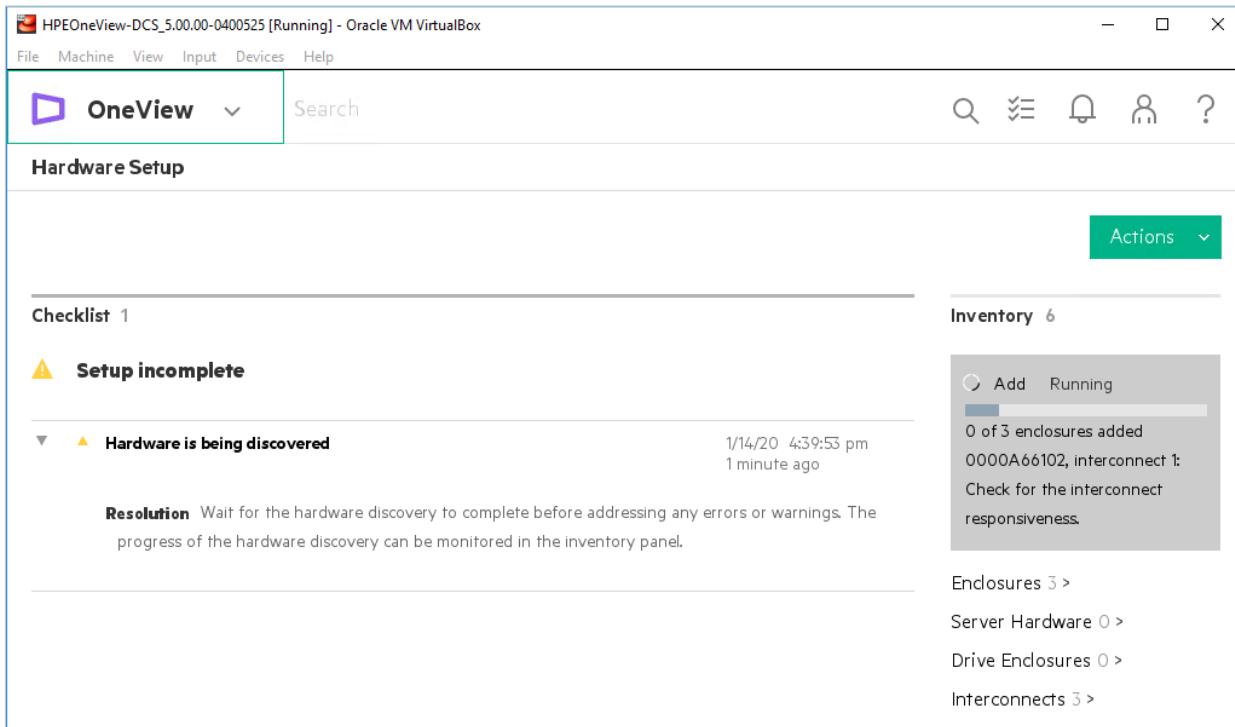
Maintenance IP address 1 active optional

Maintenance IP address 2 standby optional

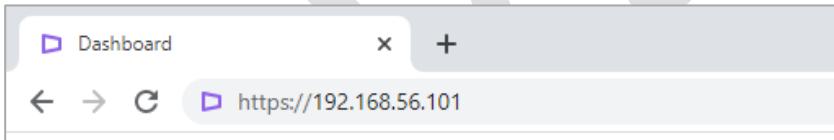
- Then click **OK**



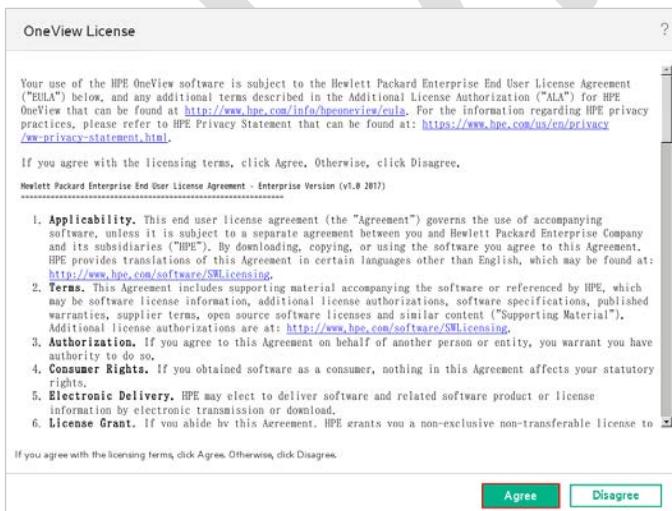
- Once network configuration completes, the initial round of hardware discovery starts:



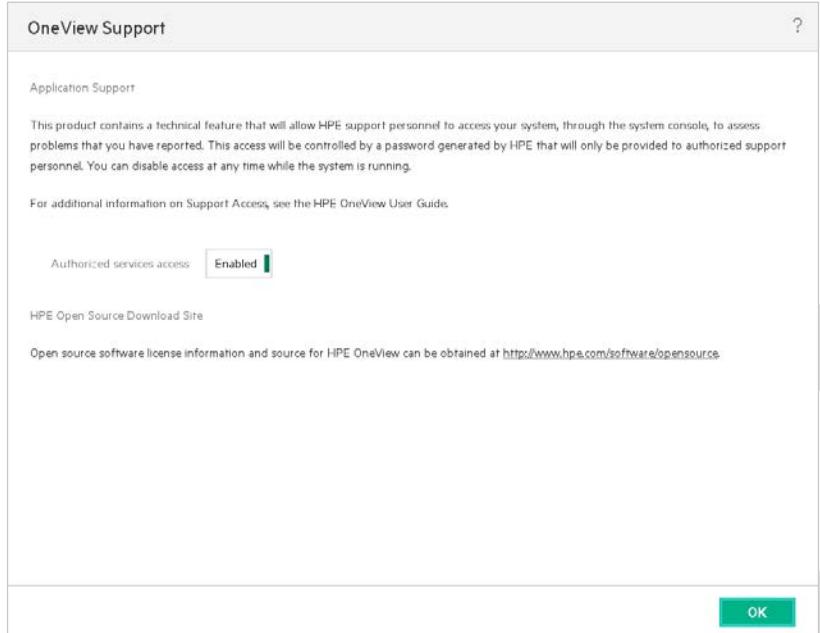
- At this point, you can access your demonstration appliance via a web browser using <https://192.168.56.101>



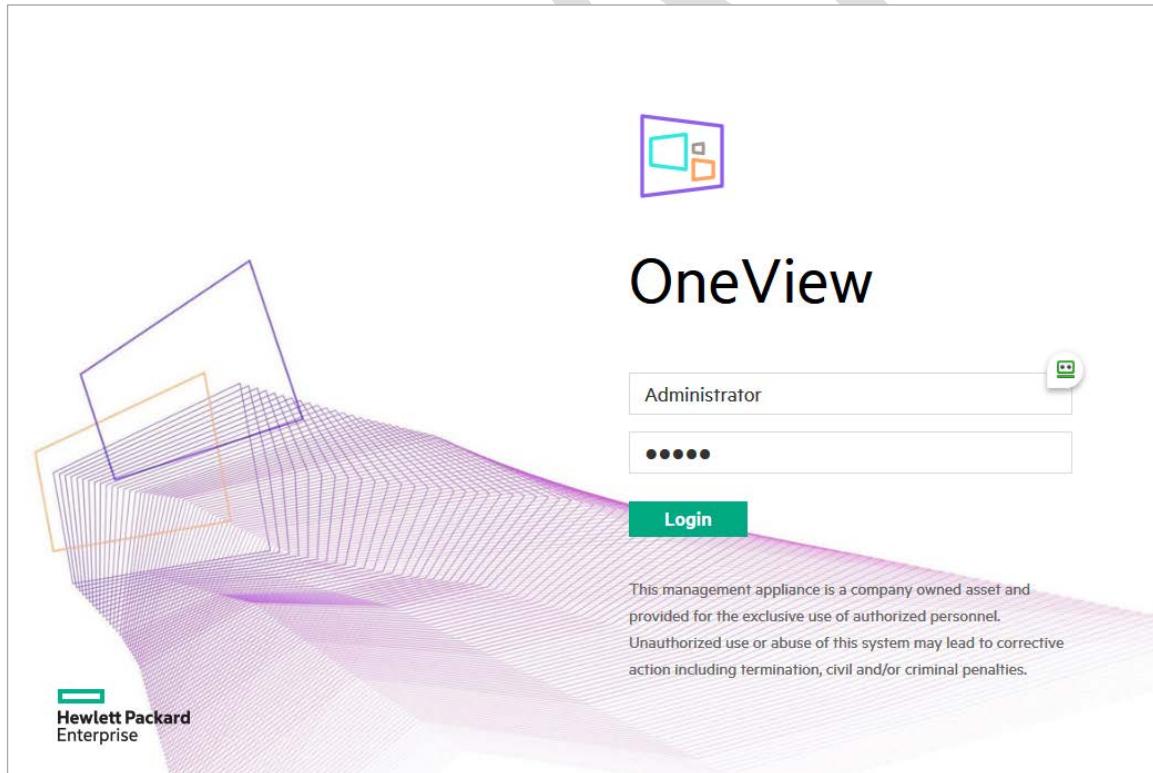
- Accept the OneView license agreement



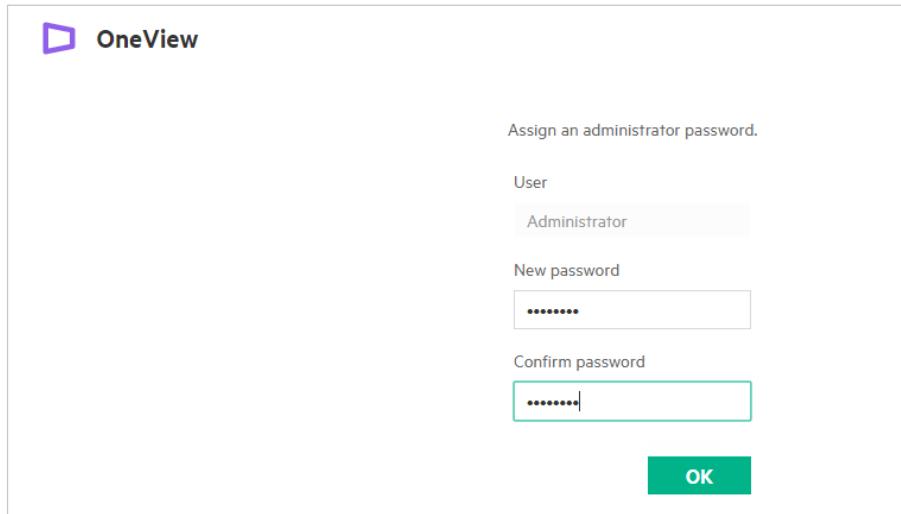
- Leave OneView Support enabled and select **OK**



- 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. At the top, there's a search bar and various navigation icons. The main area shows sections for "Server Profiles" (0) and "Server Hardware" (21). A prominent green overlay box in the center contains the "OneView Tutorial" with the message: "Welcome to the OneView user interface tutorial. This tutorial will appear the first time you log in from this browser." Below the message are buttons for "< previous", "close" (which is highlighted with a red box), "play", and "next >". At the bottom of the dashboard, there are summary counts for "Servers with profiles" (21), "No profile" (21), and "Not in maintenance" (21).

- 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-DCS_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 has a "OneView" icon, a search bar, and user icons for notifications and profile.

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 after the hardware discovery has completed. The title bar and menu are the same as the previous screenshot. The main content area displays the completed inventory statistics.

Actions

Inventory 45

- 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' at the top left. Below it is a search bar and a navigation bar with icons for search, filter, notifications, user, and help. 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 showing 45 items: Endlosures (3), Server Hardware (21), Drive Enclosures (3), and Interconnects (18). A green 'Actions' button is located in the top right corner 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:

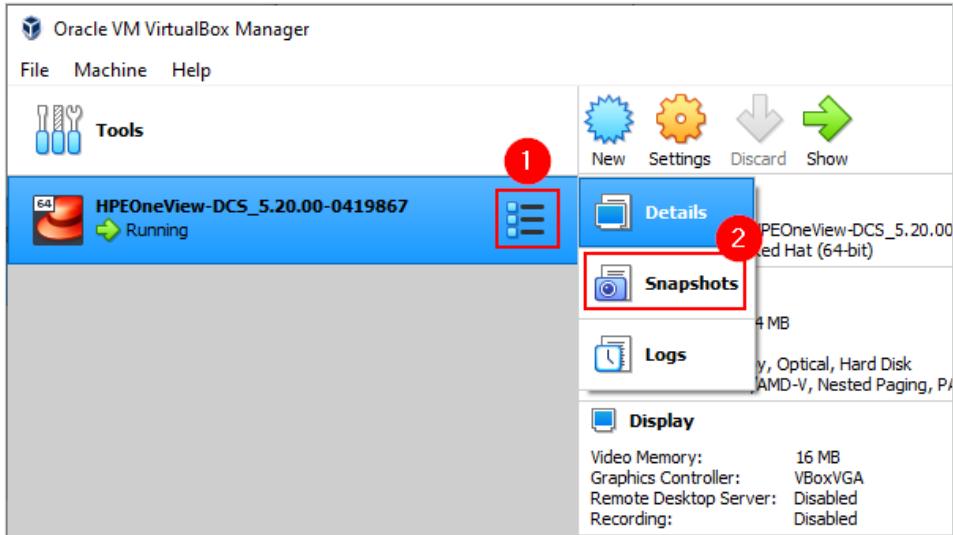
The screenshot shows the OneView interface with the title 'OneView' at the top left. Below it is a search bar with the filter 'state:running' applied. The main area is titled 'Activity' and shows a table with columns: Name, Resource, Date, State, and Owner. A message 'No matches' is displayed. To the right of the table is a 'Actions' button. A filter dropdown menu is open, showing various states: Pending (1), Running (2), Completed, Interrupted, Error, Warning, Suspended, and Cancelling. The 'Running' option is highlighted with a red border and a red number '2' indicating the count of tasks.

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

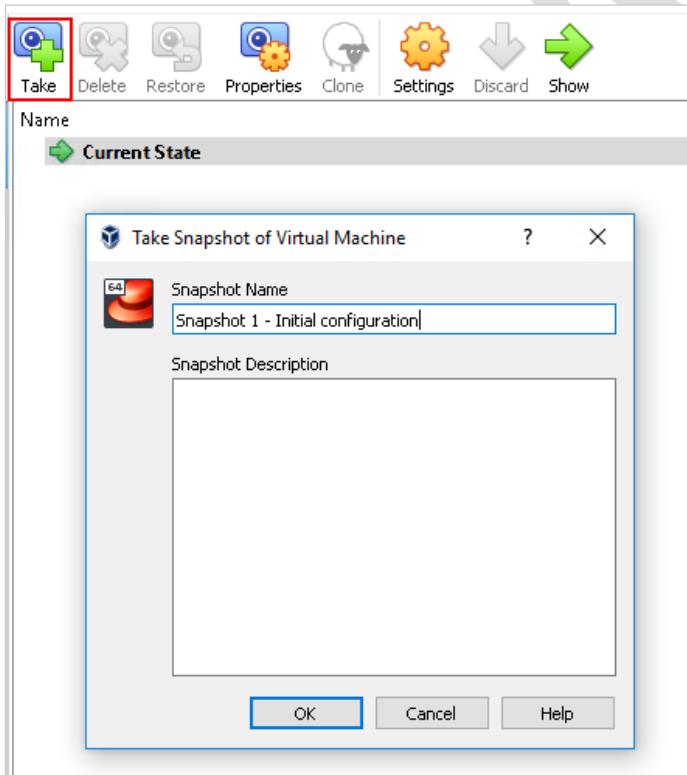


To create a snapshot with the initial configuration:

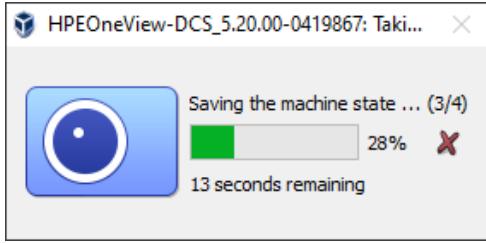
- In VirtualBox, select the **Tools** menu



- 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.

DRAFT



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.

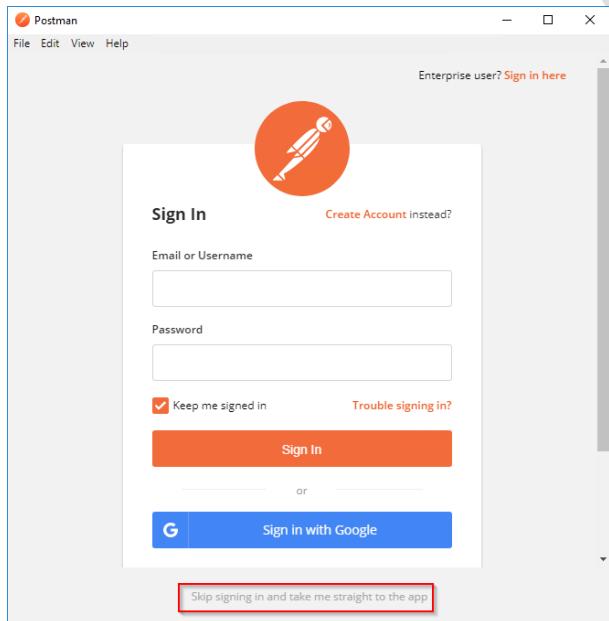
There are several simple solutions for this. My favorite is Postman as it is one of the most complete REST tools and offers useful features:

- 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

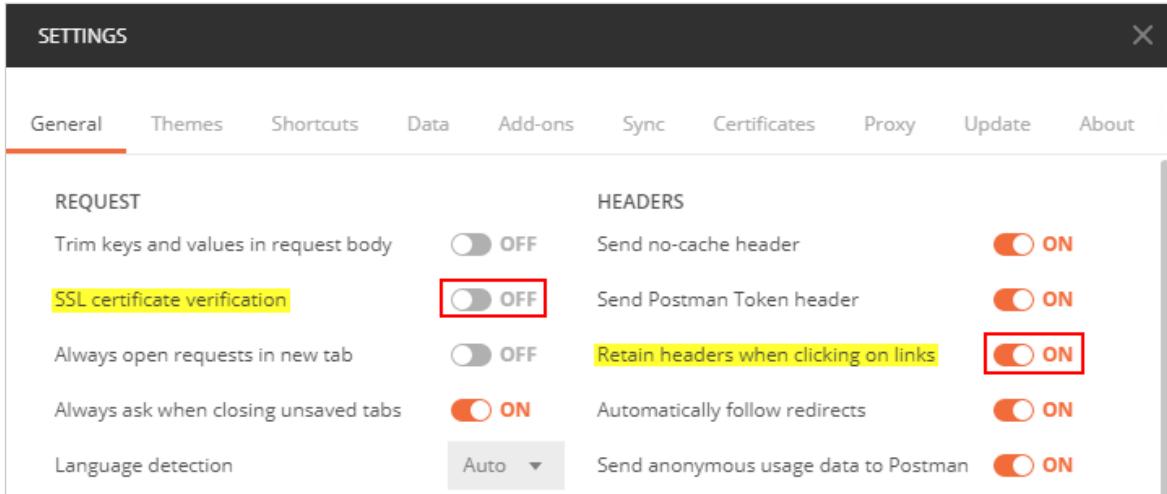
Postman can be downloaded from <https://dl.pstmn.io/download/latest/win?arch=64>

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
..	
Global Dashboard.postman_collection.json	New commit 3 days ago
OneView.postman_collection.json	New commit 3 days ago
OneView.postman_environment.json	initial commit 9 months ago
README.md	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**

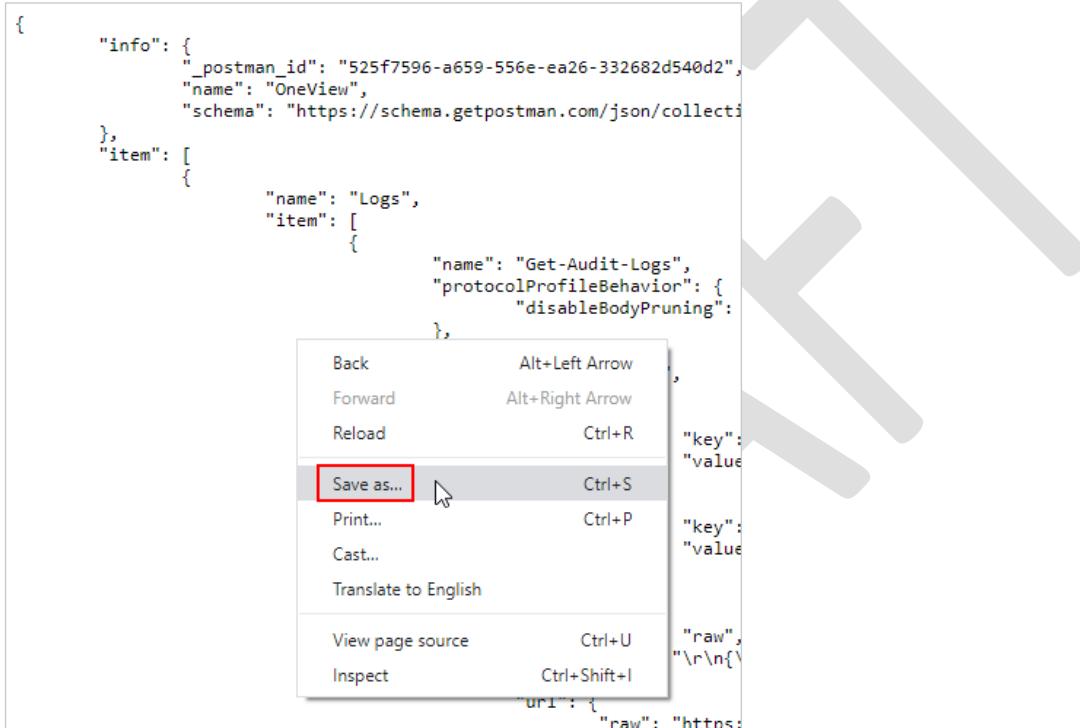


1995 lines (1995 sloc) | 43.6 KB

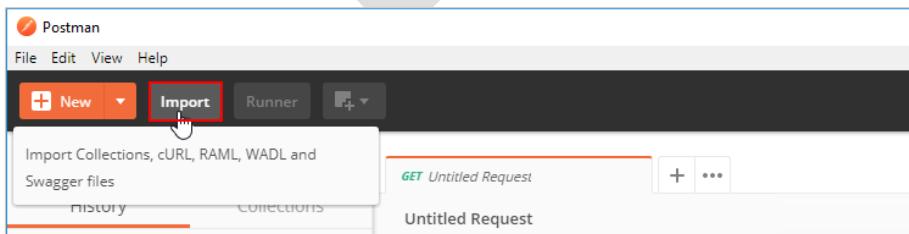
```
1 {
2     "info": {
3         "_postman_id": "525f7596-a659-556e-ea26-332682d540d2",
4         "name": "OneView",
5         "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
6     },
7 }
```

Raw Blame History

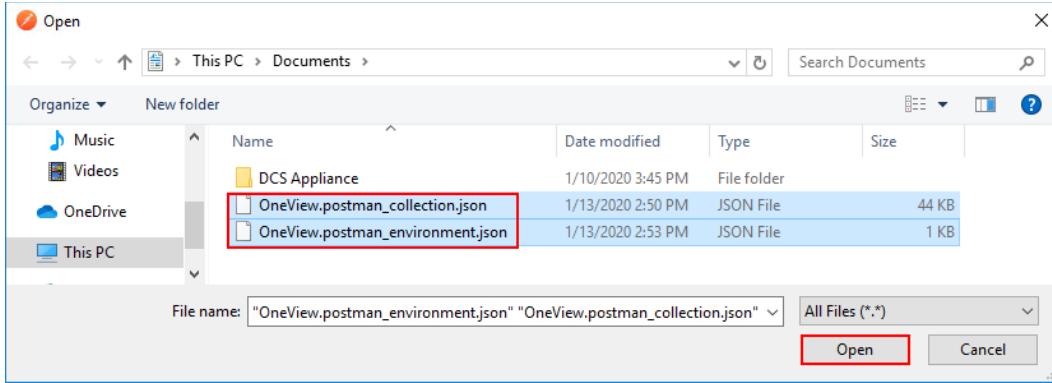
- Then right-click the page and click on **Save as...** to save the JSON file on your system:



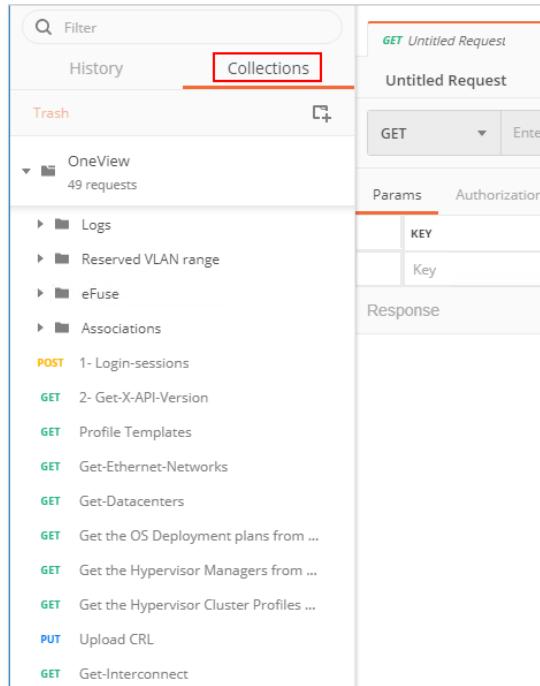
- Do the same for **OneView.postman_environment.json**.
- Back in Postman, click on **Import**:



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



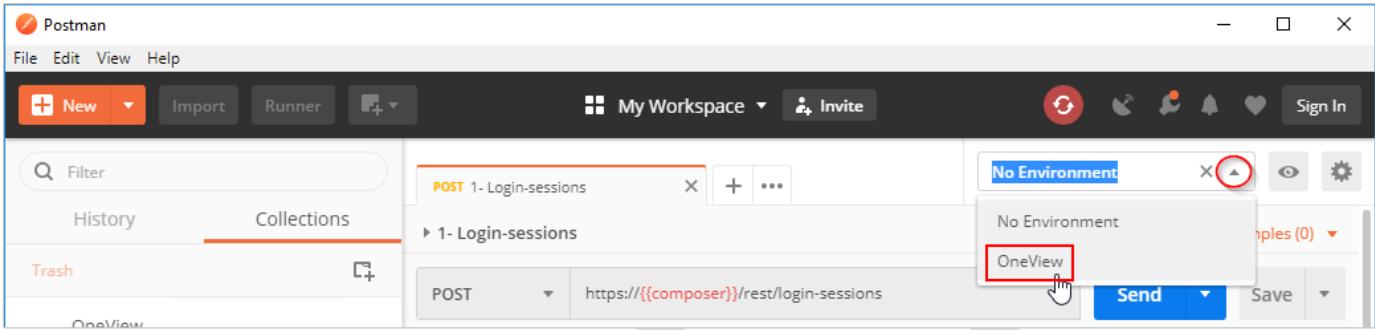
- You should see in collections, the new OneView collection



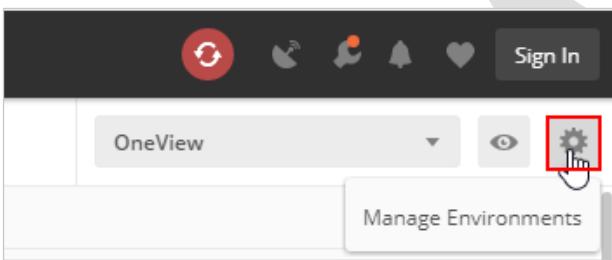
Configuration of the Postman environment

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.

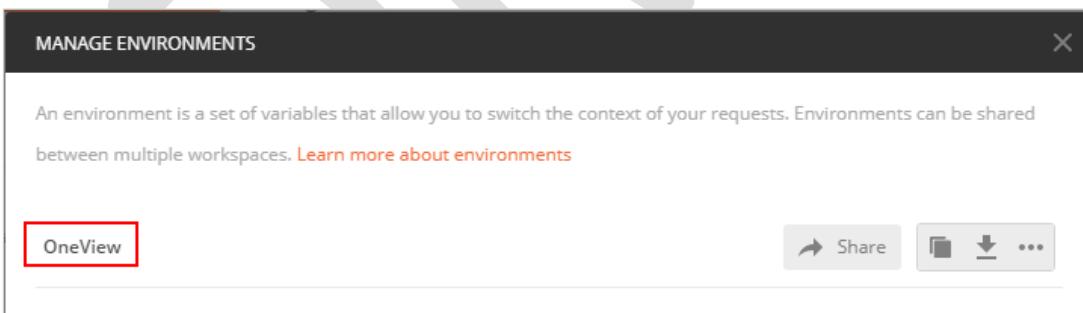
- Expand the environment menu and select **OneView**



- 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 two rows. The first row contains 'composer' with an initial value of 'composer.lj.lab' and a current value of 'composer.lj.lab'. The second row contains 'xapiversion' with an initial value of '1200' and a current value of '1200'. There are also buttons for 'Persist All' and 'Reset All'.

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<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* variable to match with our configuration.

- Change the *composer* value with **192.168.56.101**, the IP address of the appliance then press **Update** then **Close**

This screenshot shows the same 'MANAGE ENVIRONMENTS' dialog box as before, but with changes made to the 'composer' variable. The 'CURRENT VALUE' field for 'composer' now contains '192.168.56.101' (marked with a red box and number 1). The 'Update' button at the bottom right is highlighted with a red box and number 2. A red box with number 3 covers the close button in the top right corner.

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	composer	composer.lj.lab	192.168.56.101	1	X	...
<input checked="" type="checkbox"/>	xapiversion	1200	1200			
Add a new variable						

- 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 the workspace, a search bar says 'Filter' and a sidebar shows 'History' and 'Collections'. Under 'Collections', 'OneView' is expanded, showing '49 requests' and sub-items: 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. A red box highlights the 'POST 1- Login-sessions' item under 'OneView'. The main area displays a POST request for 'https://{{composer}}/rest/login-sessions'. The 'Body' tab is selected, showing a table with columns 'KEY', 'VALUE', 'DESCRIPTION', and '*** Bulk Edit'. There is one row: 'Key' (Value) and 'Value' (Description). The 'Send' and 'Save' buttons are at the bottom right.

- 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 highlighted with a red box and a circled '1'. In the body editor, the JSON code is:

```

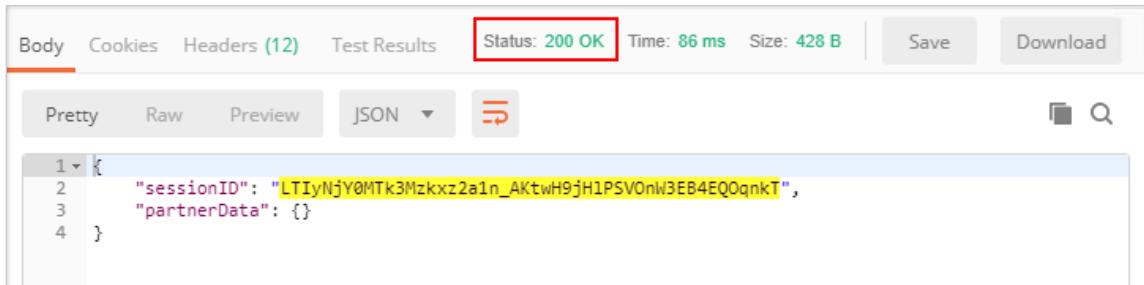
1
2 {
3   "authLoginDomain": "Local",
4   "password": "password",
5   "userName": "administrator"
6 }

```

A red box highlights the 'Save' button at the top right, which has a circled '3' above it. Other tabs like 'Params', 'Authorization', and 'Headers' are visible but not selected.

- Then press **Send**

The response we should get is the following:



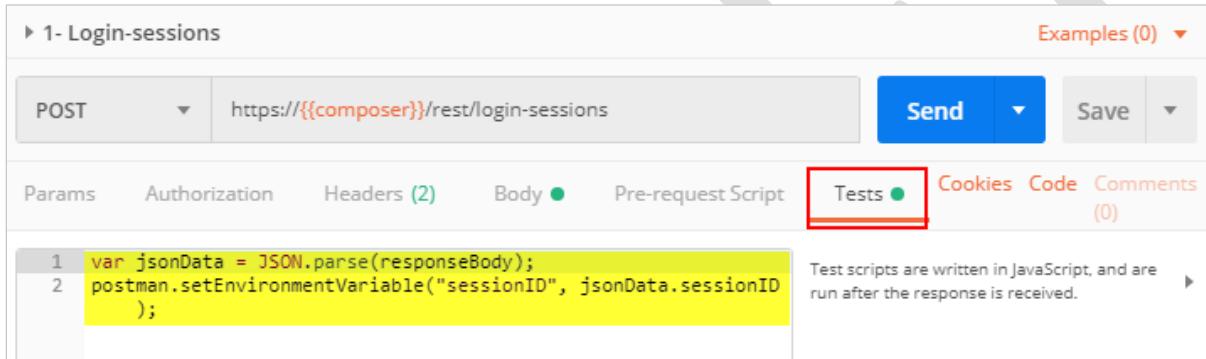
Body Cookies Headers (12) Test Results Status: 200 OK Time: 86 ms Size: 428 B Save Download

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)

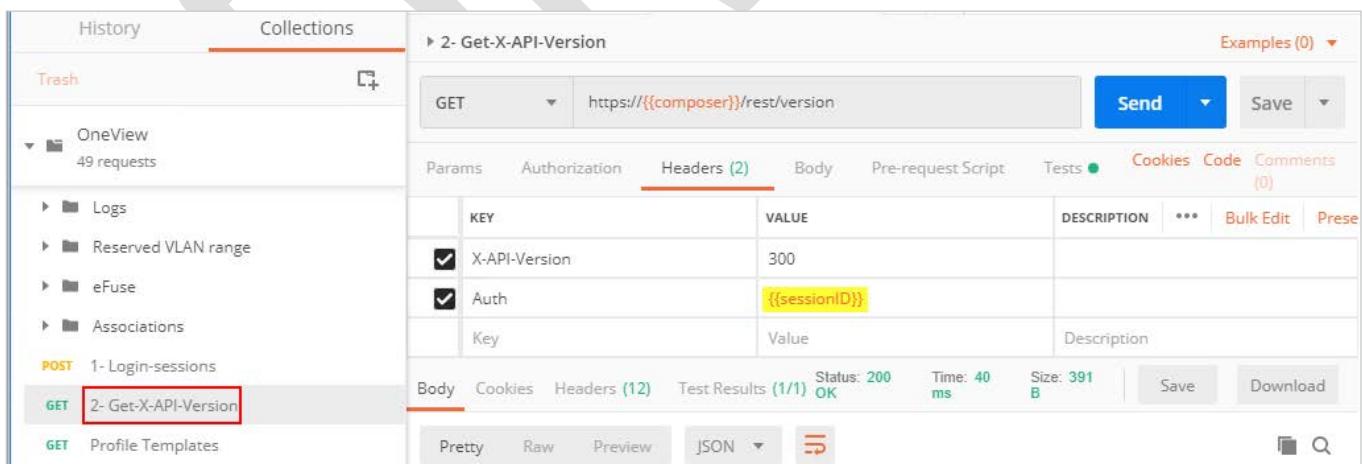
POST https://{{composer}}/rest/login-sessions Send Save

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

POST 1- Login-sessions

GET 2- Get-X-API-Version

GET Profile Templates

2- Get-X-API-Version

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
X-API-Version	300				
Auth	{{sessionID}}				
Key	Value	Description			

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 setup the HPE Synergy Composer appliance with all included hardware, we are going to use a PowerShell script.

Note: This script can also be used to show how we can fully configure a OneView/Synergy environment in front of a customer.

Dave Olker from HPE maintains a GitHub repository (<https://github.com/daveolker/Populate-HPE-Synergy-DCS>) with a powerful script to configure and populate entirely the HPE OneView DCS demonstration appliance.

Search or jump to... Pull requests Issues Marketplace Explore

daveolker / Populate-HPE-Synergy Watch 5 Star 9 Fork 6

Code Issues 0 Pull requests 1 Actions Projects 0 Wiki Security Insights

Quickly and reliably configure an HPE Synergy virtual appliance with all included hardware.

48 commits 8 branches 0 packages 0 releases 1 contributor

Branch: 5.0 ▾ New pull request Create new file Upload files Find file Clone or download ▾

daveolker Initial 5.0 Version Latest commit 1eb9239 on Sep 22, 2019

.gitattributes	Initial Version	3 years ago
.gitignore	Initial 5.0 Version	5 months ago
Cleanup_HPE_Synergy.ps1	Bug fixes	11 months ago
Populate_HPE_Synergy-Params - Sample.txt	Bug fixes	11 months ago
Populate_HPE_Synergy.ps1	Initial 5.0 Version	5 months ago
README.md	Updated Readme	3 years ago

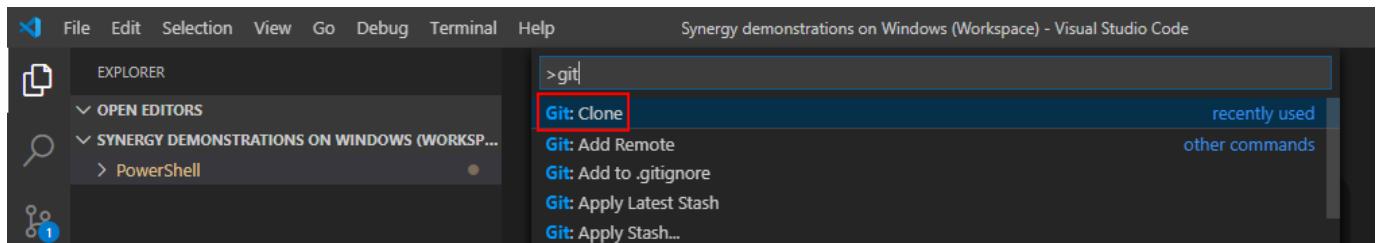
README.md

Populate HPE Synergy

To clone the Dave's repo in our VS Code workspace, we are going to use Git. One benefit of using git is that every time new scripts are pushed or changed in this repository, the VS Code Git source control will raise an alert.

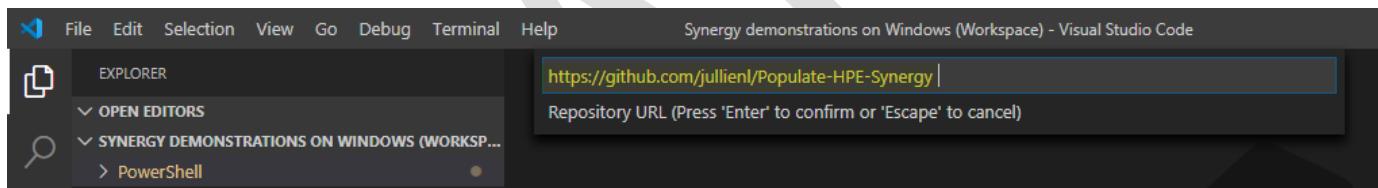
To clone the Dave's repo:

- 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:

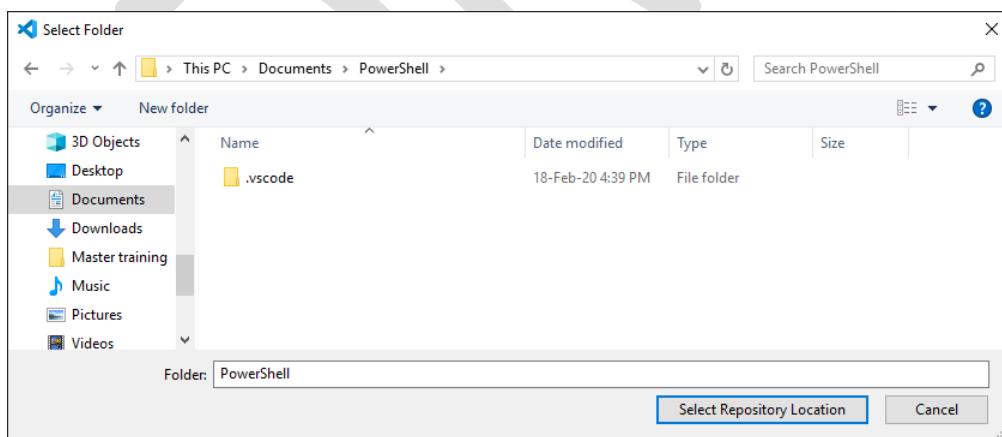


- Enter the following link: <https://github.com/jullien/Populate-HPE-Synergy>

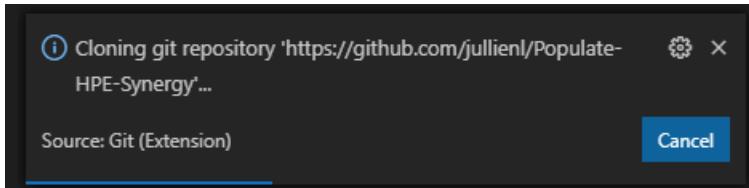
Note: This repository is forked from daveolker/Populate-HPE-Synergy. This fork provides a few changes from Dave's repository to meet the needs of our demonstration scenarios.



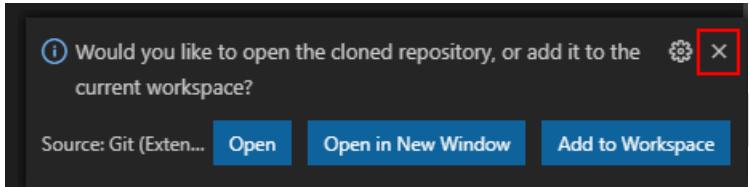
- Then select your Windows user home workspace folder as the parent directory under which to put the Dave's forked repository:



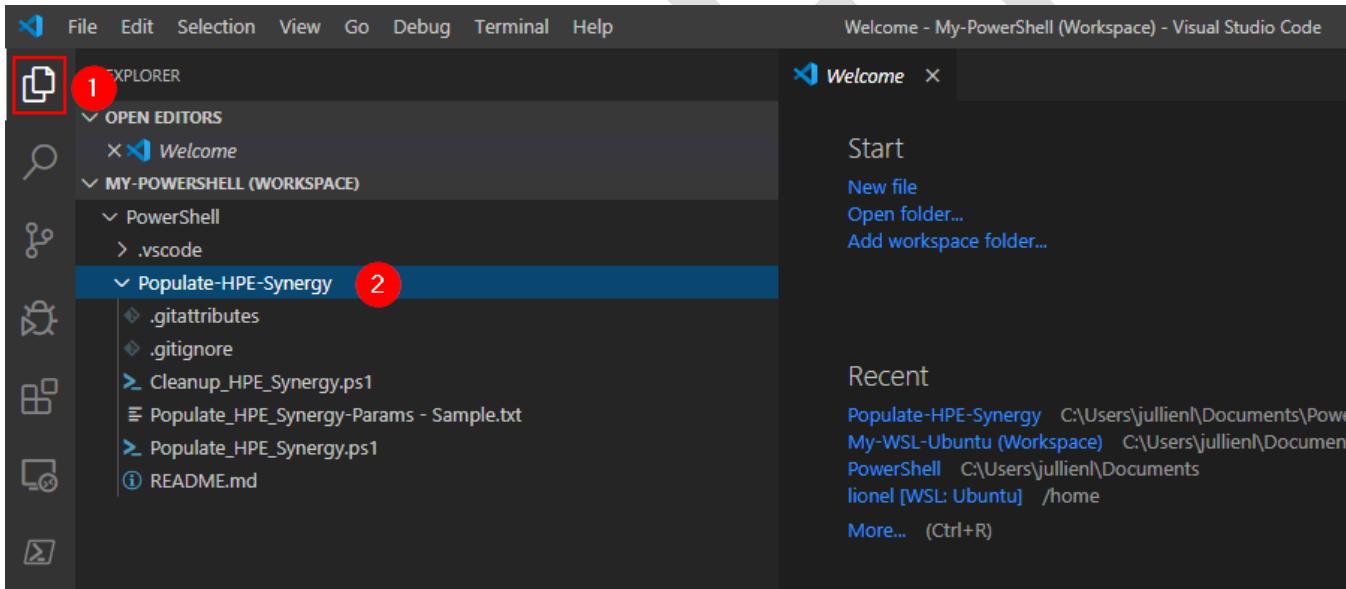
- Once selected, the cloning starts:



- Just close the pop-up window for now



- You can see now a new folder named **Populate-HPE-Synergy** in your user home directory:

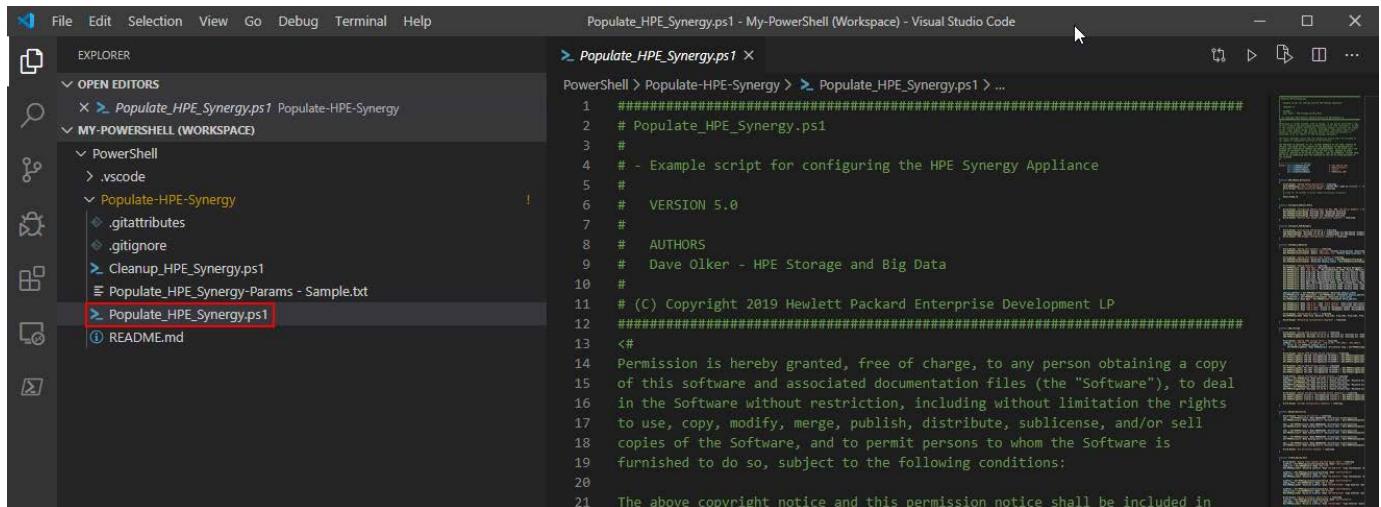


This repository provides two scripts, one to configure and populate entirely the HPE OneView DCS demonstration appliance and one to clean up everything:

`Populate_HPE_Synergy.ps1` script connects with the appliance and discovers/configures all the simulated hardware.

When the script is run, it prompts for the hostname or IP address of the Synergy appliance, the Administrator username (usually Administrator), and the Administrator password.

- Open **Populate_HPE_Synergy.ps1**



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

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Explorer Sidebar:** OPEN EDITORS (Populate_HPE_Synergy.ps1), MY-POWERSHELL (WORKSPACE) (PowerShell, .vscode, Populate-HPE-Synergy, .gitattributes, .gitignore, Cleanup_HPE_Synergy.ps1, README.md). The file `Populate_HPE_Synergy.ps1` is highlighted with a red border.
- Editor Area:** The code editor displays the contents of `Populate_HPE_Synergy.ps1`. The code includes a copyright notice and permission notice at the bottom.

```
Populate_HPE_Synergy.ps1 - My-PowerShell (Workspace) - Visual Studio Code
Populate_HPE_Synergy.ps1

PowerShell > Populate-HPE-Synergy > Populate_HPE_Synergy.ps1 > ...

1 ######
2 # Populate_HPE_Synergy.ps1
3 #
4 # - Example script for configuring the HPE Synergy Appliance
5 #
6 #   VERSION 5.0
7 #
8 #   AUTHORS
9 #   Dave Olker - HPE Storage and Big Data
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
```

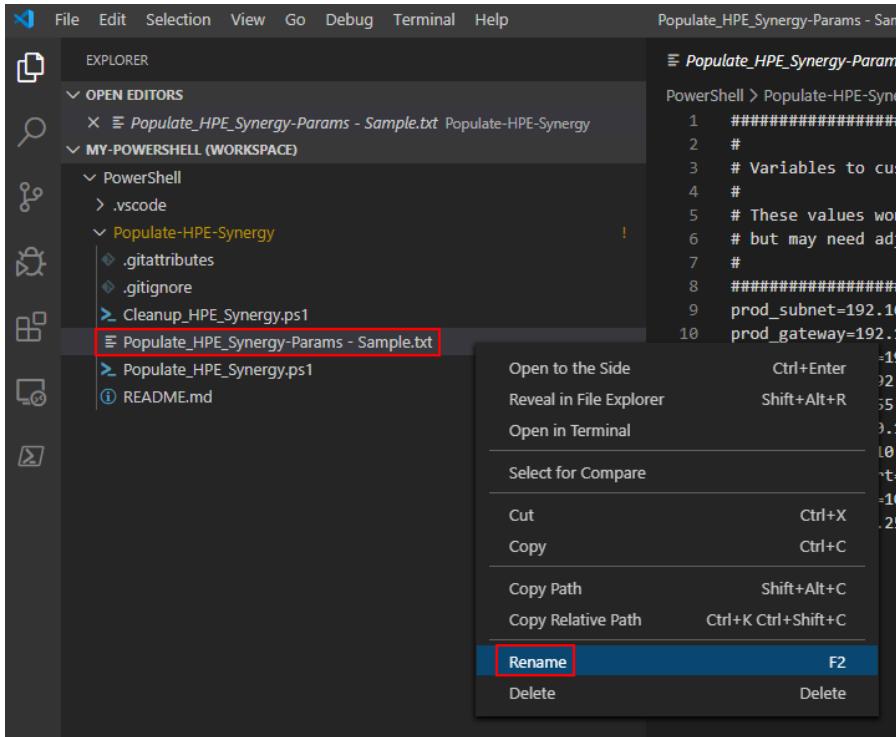
As described in the `README.md`, this script does the following:

- 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

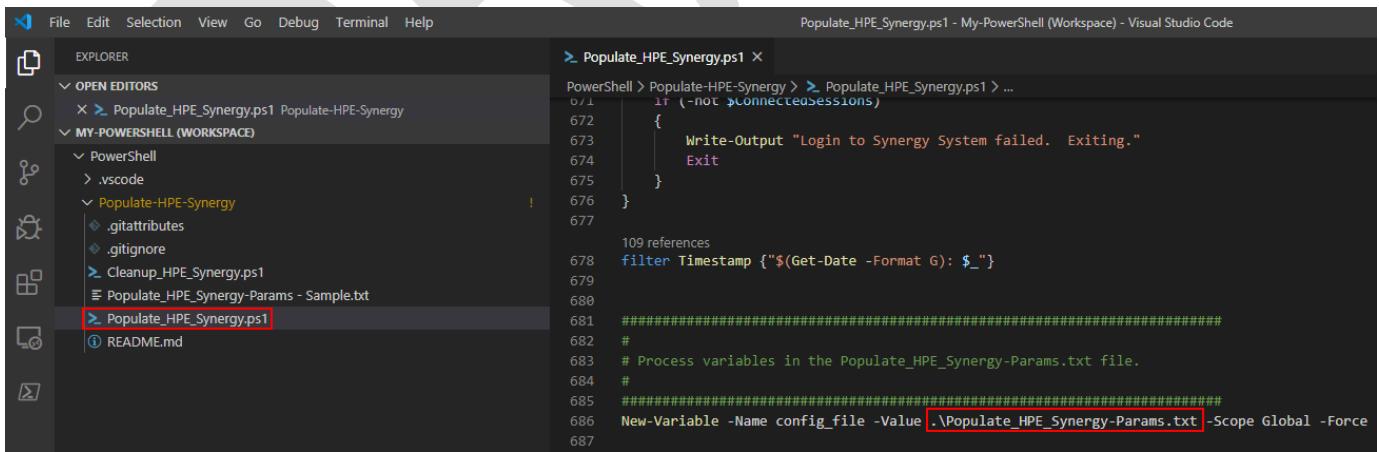
Note: To get a better display of the script, press **CTRL + K + CTRL + O** to collapse all regions

The next step is to set the `Populate_HPE_Synergy-Params - Sample.txt` used by this script to define some important variables. Good news, this configuration file is designed to work out-of-the-box with our VirtualBox using a host-only networking configuration so only one change is needed:

- Rename the file `Populate_HPE_Synergy-Params - Sample.txt` to **`Populate_HPE_Synergy-Params.txt`**



This configuration file is defined line 686 in the script:



- Scroll-down to line 700 in the script. This is where all functions defined at the beginning of the script are called:

```
698
699 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
700
701 Add_Firmware_Bundle
702 Add_Licenses
703 Configure_Address_Pools
704 Add_Remote_Enclosures
705 Rename_Enclosures
706 PowerOff_All_Servers
707 Configure_SAN_Managers
708 Configure_Networks
709 Add_Storage
710 Add_Users
711 Create_OS_Deployment_Server
712 Create_Logical_Interconnect_Groups
713 Create_Uplink_Sets
714 Create_Enclosure_Group
715 Create_Logical_Enclosure
716 Add_Scopes
717 Create_Server_Profile_Template_SY480_Gen9_RHEL_Local_Boot
718 Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
719 Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
720 Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
721 Create_Server_Profile_SY480_Gen9_RHEL_Local_Boot
722 Create_Server_Profile_SY660_Gen9_Windows_SAN_Storage
723 Create_Server_Profile_SY480_Gen9_ESX_SAN_Boot
724 Create_Server_Profile_SY480_Gen10_ESX_SAN_Boot
725
```

Each function runs a specific task:

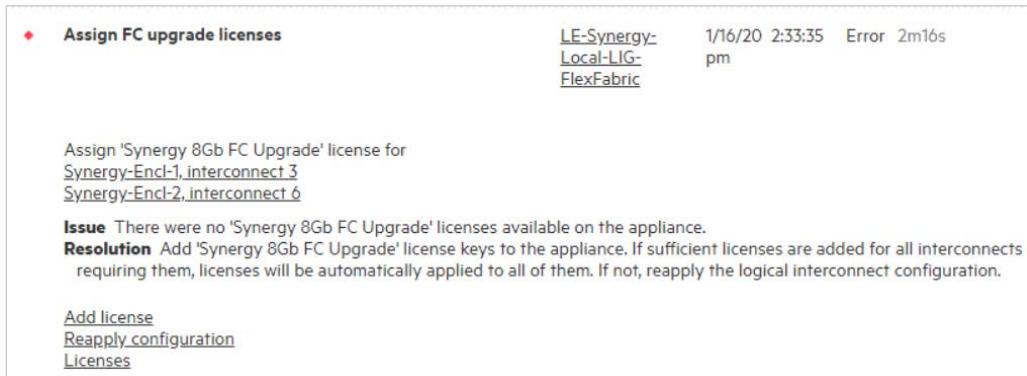
- `Add_Firmware_Bundle` adds a Service Pack for ProLiant ISO file to the OneView repository. This is optional but needed if you want to demonstrate firmware upgrade capabilities of HPE OneView but keep in mind that the firmware upgrade demonstration will be limited as you cannot update firmware of simulated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

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

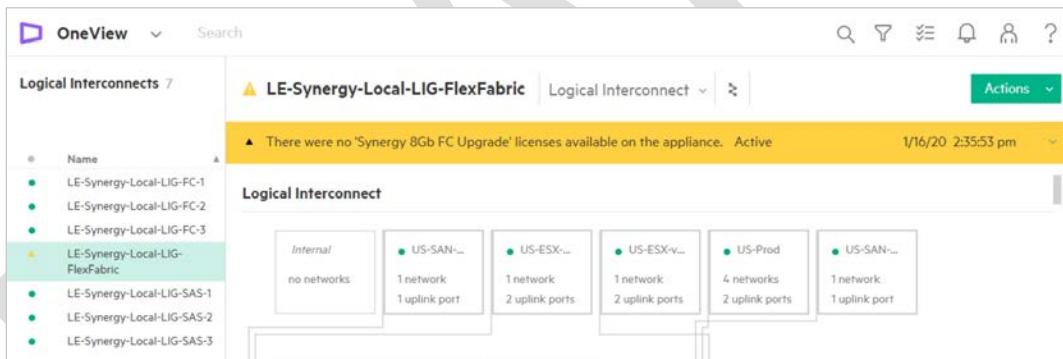
Note: You can always upload it later to your OneView appliance by going to Firmware Bundles in the Appliance section of the main menu.

When run, the function asks to specify the location of a Service Pack for ProLiant ISO file.

- `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. For Fibre Channel connectivity with the Virtual Connect SE 40 Gb F8 Module, a license is required. Without this license, FC ports cannot be activated but this is not blocking any operations other than:
 - o Throwing an error during the creation of the Logical Enclosure:



- o Displaying a warning message that there is no license for the VC Logical Interconnect:



- `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 699

```
676 }
677
678 filter Timestamp {"$(Get-Date -Format G): $_"}
679
680 #####
681 #
682 #
683 # Process variables in the Populate_HPE_Synergy-Params.txt file.
684 #
685 #####
686 New-Variable -Name config_file -Value ./Populate_HPE_Synergy-Params.txt -Scope Global -Force
687
688 if (Test-Path $config_file) {
689     Get-Content $config_file | Where-Object { !$.StartsWith("#") } | Foreach-Object {
690         $var = $.Split('=')
691         New-Variable -Name $var[0] -Value $var[1] -Scope Global -Force
692     }
693 } else {
694     Write-Output "Configuration file '$config_file' not found. Exiting." | Timestamp
695     Exit
696 }
697
698
699 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
700
701 Add_Firmware_Bundle
702 Add_Licenses
703 Configure_Address_Pools
704 Add_Remote_Enclosures
705 Rename_Enclosures
```

Activate Window
Go to Settings to activate

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

A screenshot of a code editor showing a context menu. The menu items include: Go to Definition (F12), Go to References (Shift+F12), Peek (>), Find All References (Shift+Alt+F12), Change All Occurrences (Ctrl+F2), Format Document (Shift+Alt+F), Format Document With..., Format Selection (Ctrl+K Ctrl+F), Get Help for Command (Ctrl+F1), Run Selection (F8, highlighted with a red box), Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), and Command Palette... (Ctrl+Shift+P). The code in the editor is PowerShell script, specifically a configuration script for an HPE Synergy Appliance.

```

676 }
677
678 filter Timestamp {"$(Get-Date -Format G) < "1
679
680 #####
681 ##
682 #
683 # Process variables i
684 #
685 #####
686 New-Variable -Name config
687
688 if (Test-Path $config)
689     Get-Content $config
690     $var = $_.Split([Environment]::NewLine)
691     New-Variable
692 }
693 } else {
694     Write-Output "Configuring HPE Synergy Appliance"
695     Exit
696 }
697
698 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
699
700

```

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

A screenshot of a terminal window titled "PowerShell Integrated Terminal". The output shows the following PowerShell session:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: PowerShell Integrated Terminal + ⊞ ⊛ ⌂ ⌄ ⌁ ×

Write-Output "Configuring HPE Synergy Appliance" | Timestamp
Synergy Composer IP Address [192.168.62.128]: 192.168.56.101
Administrator Username [Administrator]:
Windows PowerShell credential request.
Password required for the user 'Administrator'
Password for user Administrator: *****

ConnectionID Name UserName AuthLoginDomain Default
----- -----
1 192.168.56.101 Administrator Local True
1/17/2020 3:14:26 PM: Configuring HPE Synergy Appliance

PS C:\Users\Administrator.1j\Documents\DCS Appliance>

```

The terminal also displays a watermark for "Activate Windows" and "Go to Settings to activate Windows".

- Make sure no error is thrown.

- Then the idea is to select and run one at a time each function by pressing **F8** so that we see the output in the console. You can move back and forth between VS Code and OneView web interface to see the result of each step.

The screenshot shows a VS Code interface with a PowerShell script in the editor and its output in the terminal. The script performs various configuration tasks on an HPE Synergy Appliance. Two specific steps are highlighted: 'Add_Firmware_Bundle' (line 701) and 'F8' (line 701). A red circle with the number '1' is over 'Add_Firmware_Bundle', and another red circle with the number '2' is over 'F8'. The terminal output shows the command being run and its progress.

```
698 Write-Output "Configuring HPE Synergy Appliance" | Timestamp
699
700
701 Add_Firmware_Bundle ① ② F8
702 Add_Licenses
703 Configure_Address_Pools
704 Add_Remote_Enclosures
705 Rename_Enclosures
706 PowerOff_All_Servers
707 Configure_SAN_Managers
708 Configure_Networks
709 Add_Storage
710 Add_Users
711 Create_OS_Deployment_Server
712 Create_Logical_Interconnect_Groups
713 Create_Uplink_Sets
714 Create_Enclosure_Group
715 Create_Logical_Enclosure
716 Add_Scopes
717 Create Server Profile Template SY480 Gen9 RHEL Local Boot
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Power
1 192.168.56.101 Administrator Local True
1/17/2020 3:14:26 PM: Configuring HPE Synergy Appliance
PS C:\Users\Administrator.lj\Documents\DCS Appliance> Add_Firmware_Bundle
1/17/2020 3:18:12 PM: Adding Firmware Bundles
Optional: Specify location of Service Pack for ProLiant ISO file: []
```

Note: Only the first two steps require some input. They are optional. For `Add_Firmware_Bundle` and `Add_License` the SPP ISO file and licenses are optional, you can press **ENTER**

Note: `Add_Remote_Enclosures` takes some long minutes and is also optional. It can be skipped as we do not use the remote enclosures in the demonstration scenarios available in this guide.

- Stop after the creation of the second server profile Template line 718. The other profile creations and setup of remote enclosures will be kept for customer demonstration.

```
713 Create_Uplink_Sets
714 Create_Enclosure_Group
715 Create_Logical_Enclosure
716 Add_Scopes
717 Create_Server_Profile_Template_SY480_Gen9_RHEL_Local_Boot
718 Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
719 Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
720 Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
721 Create_Server_Profile_SY480_Gen9_RHEL_Local_Boot
722 Create_Server_Profile_SY660_Gen9_Windows_SAN_Storage
723 Create_Server_Profile_SY480_Gen9_ESX_SAN_Boot
724 Create_Server_Profile_SY480_Gen10_ESX_SAN_Boot
725
726 #
727 # Add Second Enclosure Group for Remote Enclosures
728 #
729 Create_Logical_Interconnect_Groups_Remote
730 Create_Enclosure_Group_Remote
731 Create_Logical_Enclosure_Remote
732
733 Write-Output "HPE Synergy Appliance Configuration Complete" | Timestamp
```

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

```
Disconnect-HPOVMgmt
```

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 OneView interface for managing logical enclosures. On the left, there's a sidebar with a '+ Create logical enclosure' button. The main panel is titled 'LE-Synergy-Local' and shows a yellow banner with the message: 'The logical enclosure is inconsistent with its enclosure group EG-Synergy-Local. Active'. Below this, under the 'General' section, it says 'Inconsistent with group Learn more...' and lists several items: 'EG-Synergy-Local', 'Synergy-Encl-1', 'Synergy-Encl-2', 'Synergy-Encl-3', 'LE-Synergy-Local-LIG-FlexFabric', 'LE-Synergy-Local-LIG-FC-1', 'LE-Synergy-Local-LIG-FC-3', 'LE-Synergy-Local-LIG-FC-2', 'LE-Synergy-Local-LIG-SAS-2', 'LE-Synergy-Local-LIG-SAS-1', and 'LE-Synergy-Local-LIG-SAS-3'.

- Make sure there is no OneView tasks that is still running

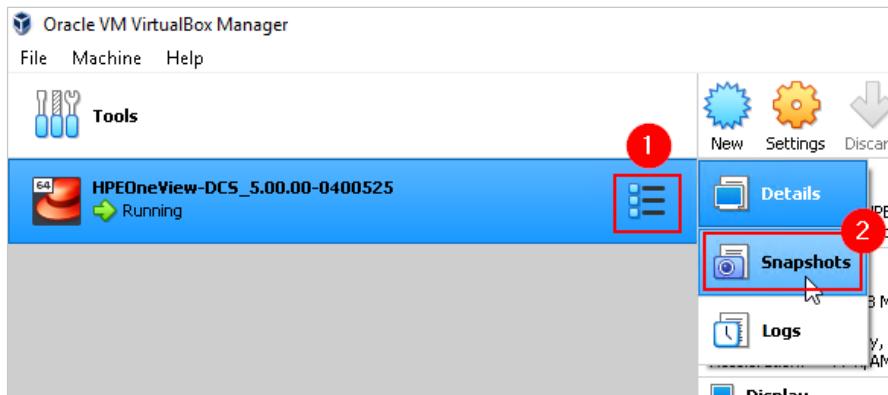
1. Go in OneView / **Activity** then use **Running** State in the Filter tool:

The screenshot shows the OneView Activity filter tool. At the top, there's a search bar with 'OneView' and a dropdown set to 'state:running'. To the right of the search bar is a filter icon with a red circle containing the number '1'. Below the search bar, the word 'Activity' is followed by '0'. The main area shows a table with columns: Name, Resource, Date, State, and Owner. A message 'No matches' is displayed. To the right of the table is a 'Actions' button. A dropdown menu is open, showing various state options: Pending (highlighted with a red box), Running (highlighted with a red box and a red circle containing '2'), Completed, Interrupted, Error, Warning, Suspended, and Cancelling.

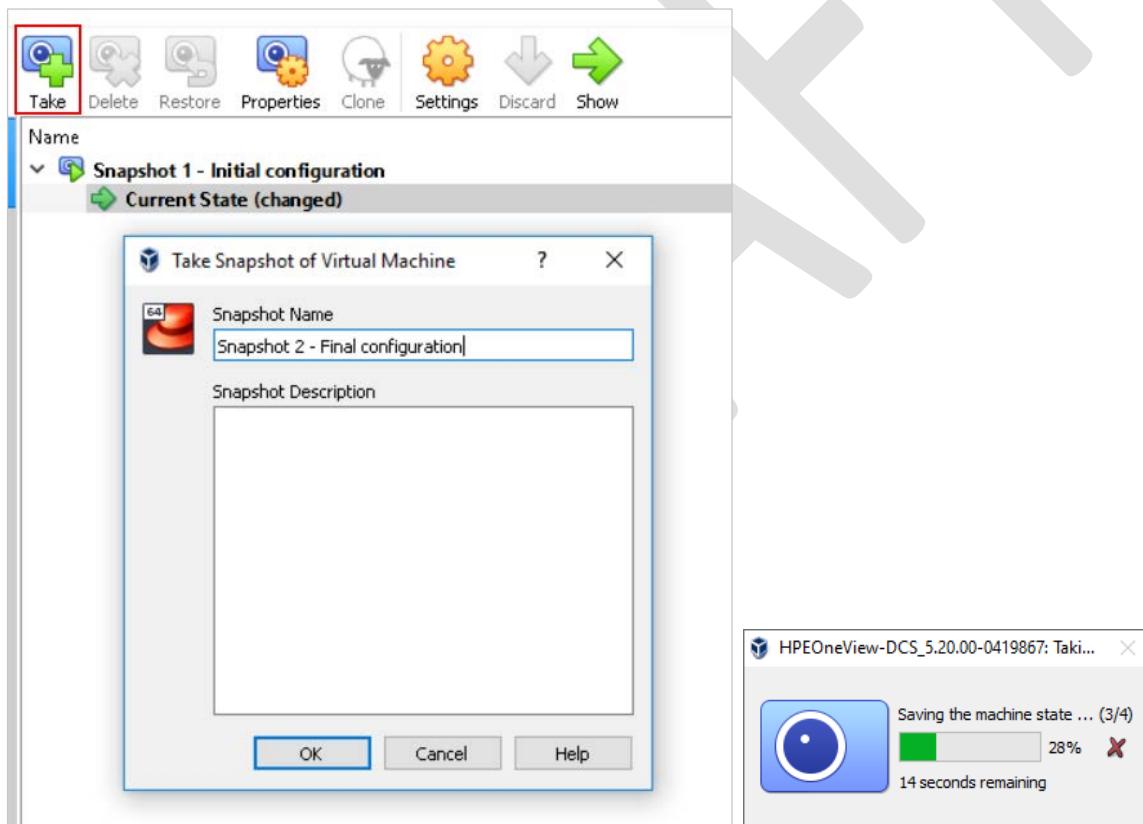
Note: Taking a snapshot while the appliance is running is a key practice to avoid waiting long minutes for the appliance to start.

To create a snapshot to save the appliance final configuration:

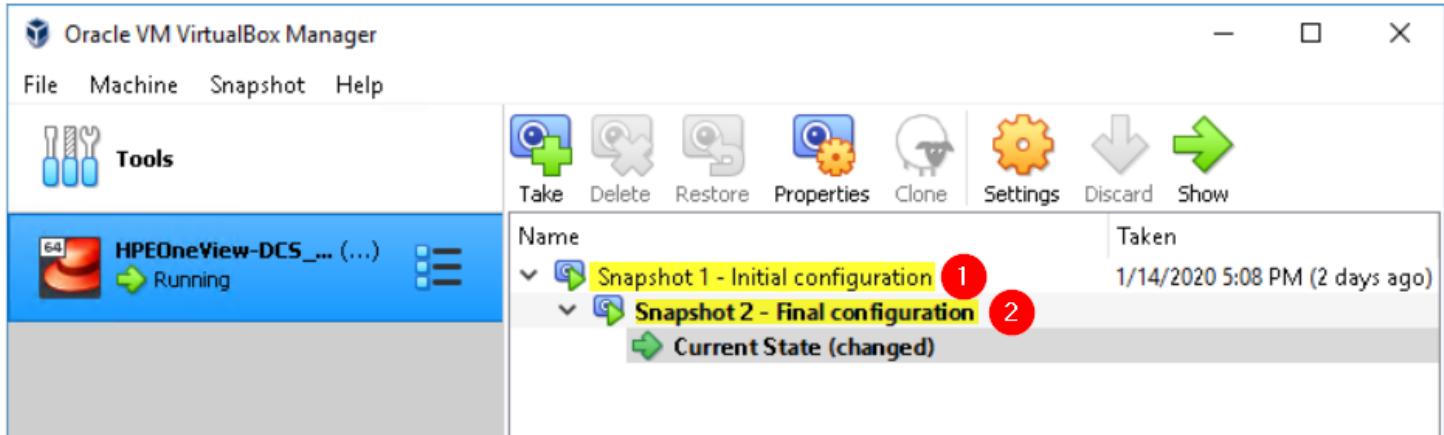
- Go to the VirtualBox interface, select the **Tools** menu then **Snapshots**



- 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...

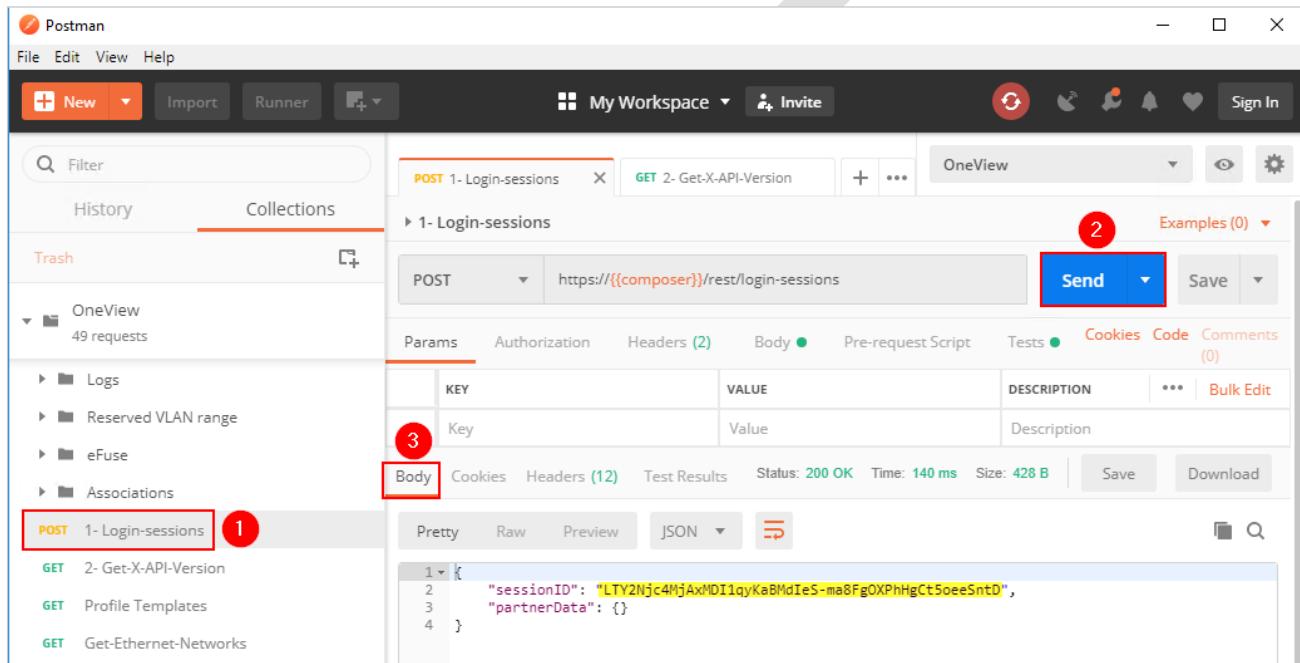


Postman - Scenario 1 – Introduction to the OneView REST API

In this scenario, we are going to use Postman to show the OneView resource model, and some typical OneView object content:

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

- Open **Postman**
- Select **1-Login-sessions** from the collection pane, then press **Send**



- Explain `sessionID` is our API authentication token that will be used to pass all other REST calls.

- Next select **Get-Interconnect** and press **Send**

The screenshot shows the Postman interface with a POST request to `https://{{composer}}/rest/login-sessions`. The 'Send' button is highlighted with a red box and a red circle with the number 1. The 'Responses' tab is selected, indicated by a red circle with the number 2. The response body is displayed in JSON format:

```

1 [
2   "sessionID": "MzA400I5NjgwNDEx5FYG340KusSw80RUMeFLBhghKcCL733q",
3   "partnerData": {}
4 ]

```

- The response displays all interconnects managed by OneView

- Click on the enclosure link of the first interconnect

```

1  {
2      "type": "InterconnectCollectionV6",
3      "uri": "/rest/interconnects/?start=0&count=5",
4      "category": "interconnects",
5      "eTag": null,
6      "created": null,
7      "modified": null,
8      "start": 0,
9      "count": 5,
10     "total": 12,
11     "prevPageUri": null,
12     "nextPageUri": "/rest/interconnects/?start=5&count=5",
13     "members": [
14         {
15             "type": "InterconnectV6",
16             "uri": "/rest/interconnects/498e7f9d-256f-49e0-8d08-5a77803cf3db",
17             "category": "interconnects",
18             "eTag": "457025d5-93b1-43b3-b652-7a6f7acd3128",
19             "created": null,
20             "modified": null,
21             "scopesUri": "/rest/scopes/resources/rest/interconnects/498e7f9d-256f-49e0-8d08-5a77803cf3db",
22             "model": "Virtual Connect SE 16Gb FC Module for Synergy",
23             "interconnectLocation": {
24                 "locationEntries": [
25                     {
26                         "type": "Enclosure",
27                         "value": "/rest/enclosures/000000000A66103"
28                     },
29                     {
30                         "type": "Bay",
31                         "value": "5"
32                     }
33                 ]
34             }
35         }
36     ]
37 }
  
```

- Notice that when doing so, Postman creates a **GET** request using this URI, press **Send**

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

- The response provides information about the enclosure UUID 000000000A66103

- Quickly explain the different components that are found in the JSON response body, an enclosure with 12 Compute bays, 6 Interconnect bays and 10 Fans, etc.

```

5   "eTag": "2020-01-14T16:56:09.961Z",
6   "created": "2020-01-14T16:40:14.481Z",
7   "modified": "2020-01-14T16:56:09.961Z",
8   "refreshState": "NotRefreshing",
9   "stateReason": "None",
10  "enclosureType": "SY12000",
11  "enclosureTypeUri": "/rest/enclosure-types/SY12000",
12  "enclosureModel": "Synergy 12000 Frame",
13  "uuid": "0000000000A66103",
14  "serialNumber": "0000A66103",
15  "partNumber": "000000-010",
16  "reconfigurationState": "NotReapplyingConfiguration",
17  "uidState": "On",
18  "licensingIntent": "NotApplicable",
19  "deviceBayCount": 12,
20  "deviceBays": [ ],
21  "interconnectBayCount": 6,
22  "interconnectBays": [ ],
23  "fanBayCount": 10,
24  "fanBays": [ ],
25  "powerSupplyBayCount": 6,
26  "powerSupplyBays": [ ],
27  "enclosureGroupUri": null,
28  "fwBaselineUri": null,
29  "fwBaselineName": null,
30  "isFwManaged": false,
31  "forceInstallFirmware": false,
32  "logicalEnclosureUri": null,
33  "managerBays": [
34    {
35      "bayNumber": 1,
36      "managerType": "EnclosureManager",
37      "uidState": "Off",
38      "bayPowerState": "Unknown",
39    }
40  ]
41  
```

Note: You can use the expand icon to unfold the different sections to get a better display

- A description of all these components is available in the REST API Reference document accessible from the Help section of OneView

Dashboard

Server Profiles 0 >

No server profiles

Server Hardware 21 >

4 Warning

Help

- Screencasts
- Tutorial**
- Documentation
- [Help on this page](#)
- [Browse help](#)
- REST API reference**
- SDK & partner program
- First time setup
- License

- In the REST API Reference document, select **Server / Enclosure / Get /rest/enclosures**

Enclosures

The enclosures resource provides REST APIs for managing enclosures. You can retrieve the enclosure resource representing any enclosure managed by the appliance, add new enclosures, and remove existing enclosures.

GET /rest/enclosures

Returns a list of enclosures matching the specified filter. A maximum of 40 enclosures are returned to the caller. Additional calls can be made to this API to retrieve any other enclosures matching the filter. Valid filter parameters include attributes of an EnclosureV7 resource.

Get a list of the enclosures with a status of OK.

Request

```
GET https://{{appl}}/rest/enclosures?filter="status='OK'"
```

Auth: abcdefghijklmnopqrstuvwxyz012345
X-Api-Version: 1200

- Scroll-down to **Response Body** and show the different component with their descriptions

Component	Description
applianceBayCount	The number of appliance bays in the enclosure.
applianceBays	A list of the appliance bays in the enclosure.
bayNumber	The bay number of the appliance.
bayPowerState	The power state of the appliance bay.

- Next you can show the Reserved VLAN range set in OneView, this information can only be accessible from the REST API, in other words, it cannot be found in the GUI. Expand **Reserved VLAN range**, select **3-Get-Reserved-VLAN-range** then press **Send**

The screenshot shows the OneView REST API interface. On the left, there's a sidebar with a tree view of requests. A node labeled "Reserved VLAN range" is expanded, and under it, a request labeled "GET 3-Get-Reserved-VLAN-range" is selected. This request is highlighted with a red box and has a red number "2" next to it. At the top right, there's a toolbar with various buttons, one of which is "Send". This "Send" button is also highlighted with a red box and has a red number "3" next to it.

```

1
2   "type": "FabricCollectionV2",
3   "uri": "/rest/fabrics/?start=0&count=1",
4   "category": "fabrics",
5   "eTag": null,
6   "created": null,
7   "modified": null,
8   "start": 0,
9   "count": 1,
10  "total": 1,
11  "prevPageUri": null,
12  "nextPageUri": null,
13  "members": [
14    {
15      "type": "fabricV2",
16      "uri": "/rest/fabrics/390d529e-159c-4960-8d9c-8741138e5047",
17      "category": "fabrics",
18      "eTag": "ca567f7b-6841-4835-85b2-27bfa0d3bc8",
19      "created": "2020-01-10T15:21:09.593Z",
20      "modified": "2020-01-10T15:21:10.262Z",
21      "domainUri": "/rest/domains/6990658e-216c-42de-ab0d-ab5d90d04433",
22      "reservedVlanRange": {
23          "type": "vlan-pool",
24          "uri": "/rest/fabrics/390d529e-159c-4960-8d9c-8741138e5047/reserved-vlan-range",
25          "category": "reserved-vlan-range",
26          "eTag": null,
27          "created": "2020-01-10T15:21:09.593Z",
28          "modified": "2020-01-10T15:21:10.262Z",
29          "start": 3967,
30          "length": 128
31      },
32      "fabricType": "Default",
33      "foreignState": "NotApplicable",
34      "foreignManager": null,
35      "refreshState": "NotApplicable",
36      "description": null,
37      "state": "NotApplicable",
38      "status": "OK",
39      "name": "DefaultFabric"

```

Note: There is a reserved VLAN pool, a range of VLANs used for Tunnel, Untagged and Native FC networks. These VLAN IDs are reserved and cannot be used. 128 is the default reserved range [3967-4094]. The minimum size of the pool must be 60 VLANs [4035-4094] to ensure the pool is not exhausted. The pool can only be reduced using the REST API.

Note: The OneView PowerShell library provides a cmdlet to change the Reserved VLAN range:
`Set-HPOVRReservedVlanRange -start 4035 -Length 60`

- To demo how to reduce the reserved range, you can run the next REST query in the list: **4- Put-Change-Reserved-VLAN-range** then select **Body**

The screenshot shows the Postman interface with the 'Collections' tab selected. On the left, under the 'OneView' collection, the 'Reserved VLAN range' folder contains several requests: '1- Login-sessions' (POST), '2- Get-X-API-Version' (GET), '3- Get-Reserved-VLAN-range' (GET), '4- Put-Change-Reserved-VLAN-range' (PUT, highlighted with a red box labeled '1'), and '5- Get-Task-object-Result' (GET). On the right, the '4- Put-Change-Reserved-VLAN-range' request is expanded. The 'Body' tab is selected (highlighted with a red box labeled '2'). The JSON payload is:

```

1 {
2   "start": 3968,
3   "length": 127,
4   "type": "vlan-pool"
5 }

```

- This example shows how to reduce the reserved VLAN to 127 VLANs and set the reserved range to [3968-4094] releasing VLAN 3967 for other use like in a Cisco ACI environment.

Note: VLAN 3967 is a recommended Cisco choice for the ACI infrastructure VLAN

- Click **Send** to modify the reserved range. You should not see any Body Response. This is as expected. This request does not return any Body but a Headers response with a task ID.

The screenshot shows the 'Headers' tab for the PUT request. The headers listed are:

- Date → Wed, 15 Jan 2020 21:12:04 GMT
- Server → Apache
- Location → <https://192.168.56.101/rest/tasks/49fd2926-327a-45bd-94db-93a3f965ff78> (highlighted with a red box)
- Cache-Control → no-cache
- Pragma → no-cache

- Select the next REST call in the list **5- Get-Task-object-Result** to query this task ID, press **Send**

The screenshot shows the Postman application interface. On the left, the 'Collections' tab is selected, displaying a list of API requests under the 'OneView' collection. The request '5- Get-Task-object-Result' is highlighted with a red box and a circled '1'. On the right, the details panel shows the selected request: a GET method with the URL {{taskresult}}. The 'Send' button is highlighted with a red box and a circled '2'. Below the URL, the 'Params' tab is active, showing a single parameter 'Key' with 'Value'. The 'Body' tab is also visible, containing JSON response code.

```

1  {
2   "type": "TaskResourceV3",
3   "uri": "/rest/tasks/49fd2926-327a-45bd-94db-93a3f965ff78",
4   "category": "tasks",
5   "eTag": "1",
6   "created": "2020-01-15T21:12:04.710Z",
7   "modified": "2020-01-15T21:12:05.231Z",
8   "taskStatus": null,
9   "taskState": "Completed",
10  "owner": "Administrator",
11  "parentTaskUri": null,
12  "userInitiated": true,
13  "associatedTaskUri": null,
14  "name": "Update Fabric Reserved Range",
15  "taskErrors": [],
16  "taskOutput": [],
17  "progressUpdates": [],
18  "totalSteps": 0,
19  "completedSteps": 0,
20  "percentComplete": 100,
21  "expectedDuration": 0,
22  "computedPercentComplete": 100,
23  "data": {
24    "task-category": "GenericEdit"
25  },
26  "taskType": "User",
27  "stateReason": null,
28  "associatedResource": {
29    "resourceName": "DefaultFabric",
30    "resourceUri": "/rest/fabrics/390d529e-159c-4960-8d9c-8741138e5047",
31    "resourceCategory": "fabrics",
32    "associationType": "MANAGED_BY"
33  },
34  "hidden": false,
35  "isCancellable": false,
36  "startTime": "2020-01-15T21:12:04.723Z"
37 }

```

- You should see the task completion in the response body

- Select step **6- Get-New-Reserved-VLAN-range** call then press **Send**

The screenshot shows the Postman application interface. On the left, the sidebar lists collections and requests. A specific request titled "6- Get-New-Reserved-VLAN-range" is highlighted with a red box and a circled '1'. On the right, the main panel shows the request details: method "GET", URL "https://{{composer}}/{{fabricid}}/reserved-vlan-range", and the "Send" button highlighted with a red box and a circled '2'. The response body is displayed in JSON format:

```
1  {
2   "type": "vlan-pool",
3   "uri": "/rest/fabrics/390d529e-159c-4960-8d9c-8741138e5047/reserved-vlan-range",
4   "category": "reserved-vlan-range",
5   "eTag": null,
6   "created": "2020-01-10T15:21:09.593Z",
7   "modified": "2020-01-15T21:12:05.216Z",
8   "start": 3968,
9   "length": 127
10 }
```

- You should see the new reserved range starting now at VLAN 3968 as requested

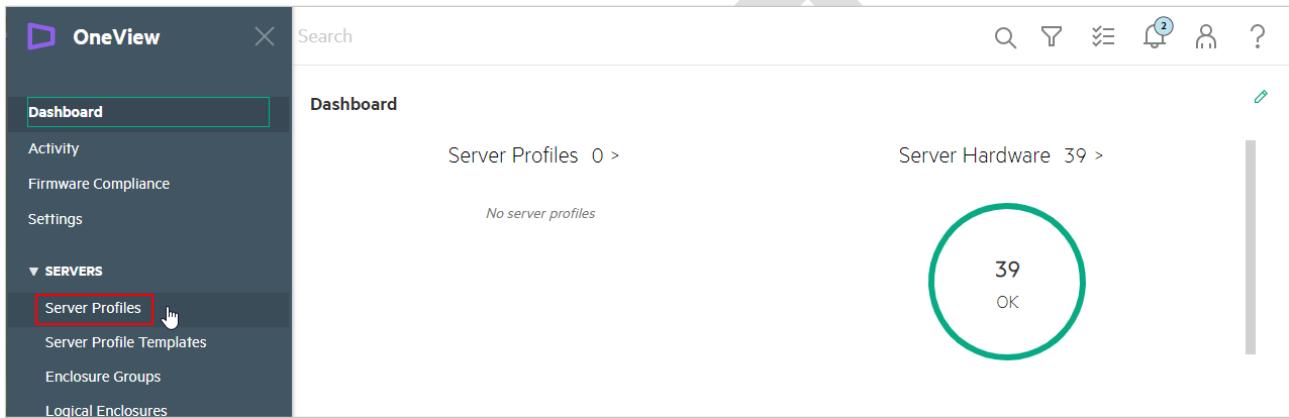
This concludes Postman – scenario 1

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 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**:

Create Server Profile General ?

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.

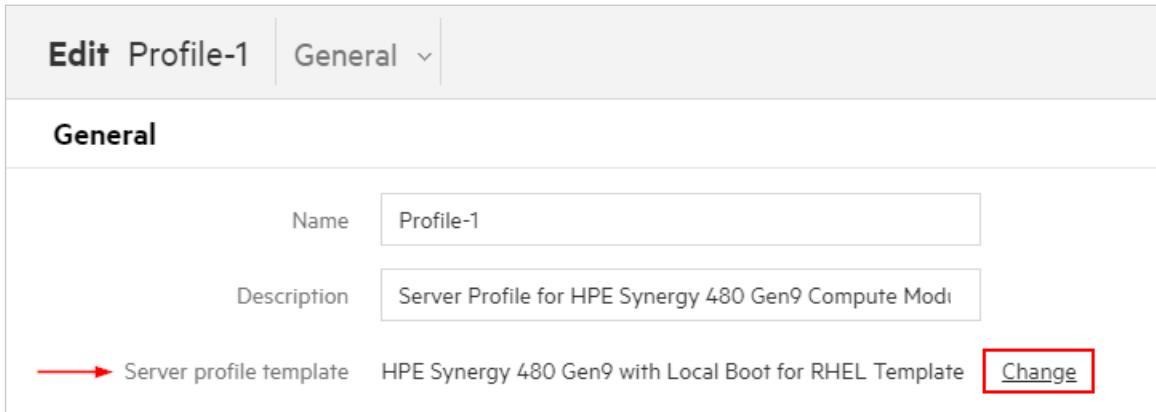
General

Name	Profile-1
Scope	Select zero or more scopes
Server profile template	HPE Synergy 480 Gen9 with Local Boot for RHEL Temp X
Description	Server Profile for HPE Synergy 480 Gen9 Compute Module w
Server hardware	Synergy-Encl-1, bay 5 X
<input type="checkbox"/> Show empty bays	
Server hardware type	SY 480 Gen9 1
Enclosure group	EG-Synergy-Local
Affinity	Device bay

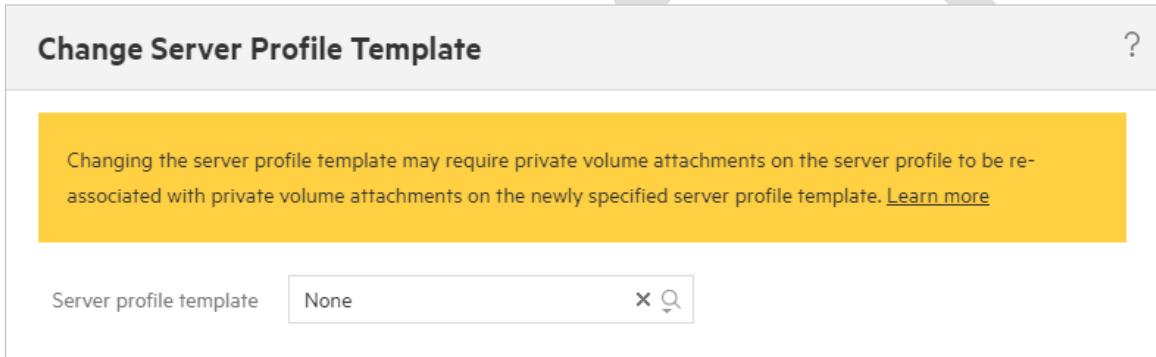
- Then click **Create**

Once created, it is necessary to unlink the server Profile from the Server Profile Template to get the most from the `ConvertTo-HPOVPowerShellScript` 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
- Copy/paste the following commands in the console to connect to OneView:

```
#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-HPOVMgmt -appliance $IP -Credential $credentials
```

Note: You can open the PowerShell console using **CTRL + SHIFT + `**



- A OneView connection confirmation should be displayed:

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> #IP address of OneView
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $IP = "192.168.56.101"
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy>
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> # OneView Credentials
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $username = "Administrator"
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $password = "password"
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy>
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy>
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> Connect-HPOVMgmt -appliance $IP -Credential $credentials

This management appliance is a company owned asset and provided for the exclusive use of authorized personnel. Unauthorized use or abuse of this system
n, civil and/or criminal penalties.

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

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> []
```

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

```
get-hpovserverprofile -name Profile-1
```

```
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> get-hpovserverprofile -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\Populate-HPE-Synergy> []
```

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

```
Get-HPOVServerProfile -Name Profile-1 | ConvertTo-HPOVPowerShellScript
```

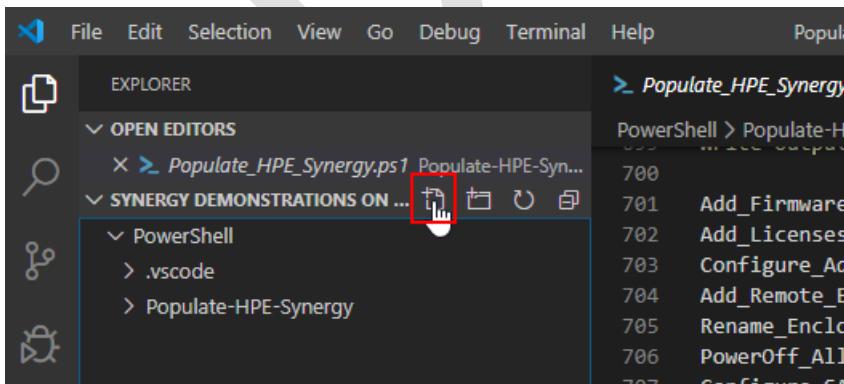
Explain to the customer that this cmdlet can assist administrators or scripters to help generate script code from specific resources. The code generated is a starting point to be used for repeating similar tasks performed by the UI, or to incorporate into scripts or workflows.

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

- Lastly, disconnect the PowerShell client to the appliance:

Disconnect-HPOVMgmt

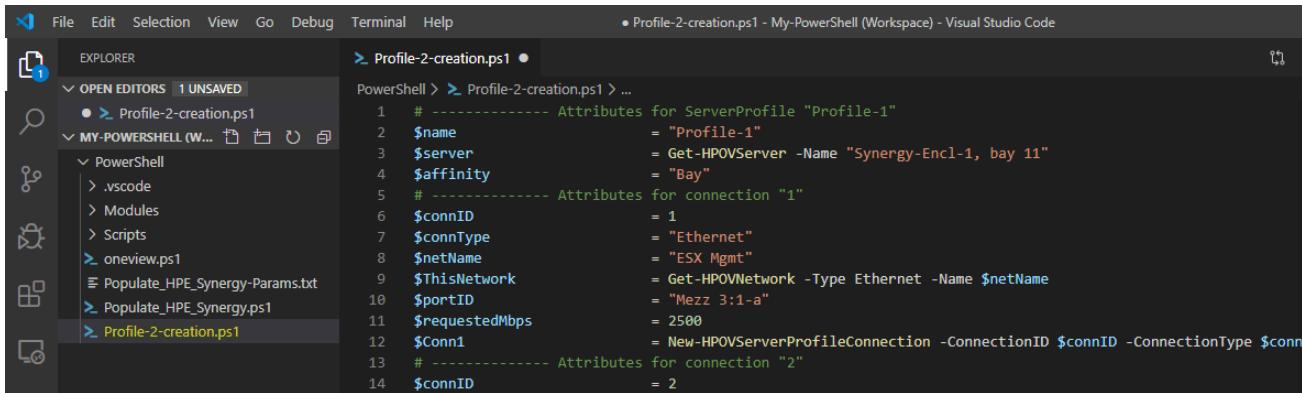
- Next, we can demonstrate how we can easily create a script using this generated code to create a new Server Profile in OneView. This will require just a few modifications. Click on **New File** to create a new script:



- 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-HPOVPowerShellScript` in this new PowerShell script



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

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Terminal:** Profile-2-creation.ps1 - My-PowerShell (Workspace) - Visual Studio Code
- Explorer:** Shows 1 UNSAVED file: Profile-2-creation.ps1. It also lists other files and folders under MY-POWERSHELL (W...), including .vscode, Modules, Scripts, oneview.ps1, Populate_HPE_Synergy-Params.txt, Populate_HPE_Synergy.ps1, and Profile-2-creation.ps1.
- Code Editor:** Displays the PowerShell script content:

```
PowerShell > > Profile-2-creation.ps1 > ...
1 # ----- Attributes for ServerProfile "Profile-1"
2 $name = "Profile-1"
3 $server = Get-HPOVServer -Name "Synergy-Encl-1, 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 = 2500
12 $Conn1 = New-HPOVServerProfileConnection -ConnectionID $connID -ConnectionType $connType
13 # ----- Attributes for connection "2"
14 $connID = 2
```

- 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-HPOVMgmt -appliance $IP -Credential $credentials
```

- Then change the following values:

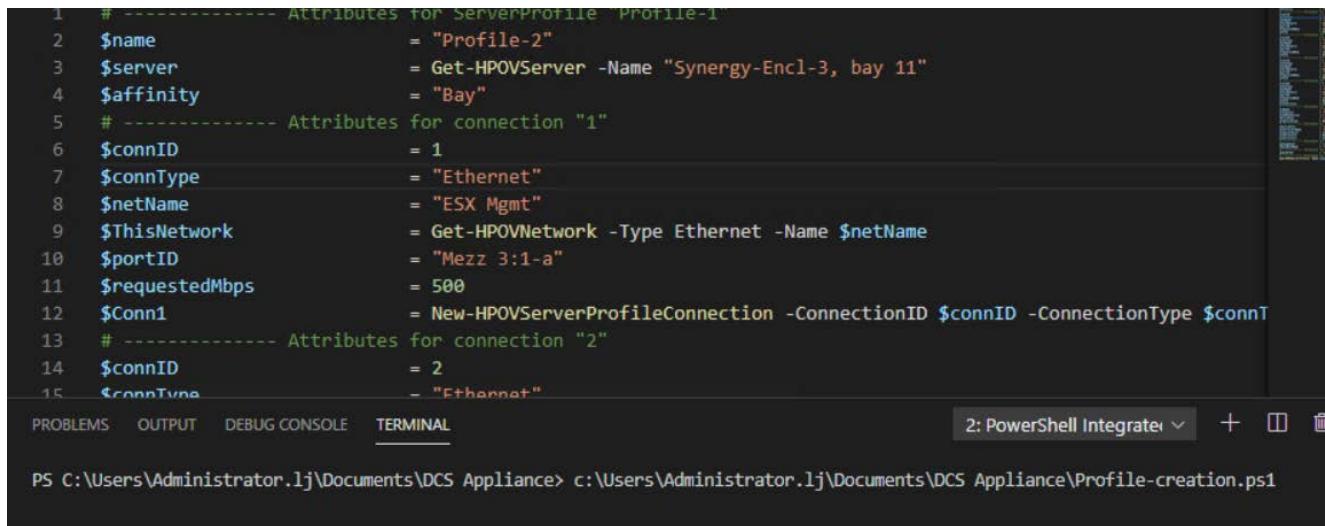
- **\$name with Profile-2**
- **\$server with Synergy-Encl-2, bay 5**
- The requested bandwidth of connection **1** and **2** to **7500** Mbps

```
➤ Populate_HPE_Synergy.ps1 • ➤ Profile-2-creation.ps1 •
PowerShell > ➤ Profile-2-creation.ps1 > ...
4  # OneView Credentials
5  $username = "Administrator"
6  $password = "password"
7
8  $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
9  $credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
10
11 Connect-HPOVMgmt -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 RHEL"
16 $server        = Get-HPOVServer -Name "Synergy-Encl[2] bay 5"
17 $affinity      = "Bay"
18 # ----- Connections section
19 # ----- Attributes for connection "1"
20 $connID        = 1
21 $connName      = "Mgmt-1"
22 $connType      = "Ethernet"
23 $netName       = "Mgmt"
24 $ThisNetwork   = Get-HPOVNetwork -Type Ethernet -Name $netName
25 $portID        = "Mezz 3:1-a"
26 $requestedMbps = 7500
27 $Conn1         = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy...
28 # ----- Attributes for connection "2"
29 $connID        = 2
30 $connName      = "Mgmt-2"
31 $connType      = "Ethernet"
32 $netName       = "Mgmt"
33 $ThisNetwork   = Get-HPOVNetwork -Type Ethernet -Name $netName
34 $portID        = "Mezz 3:2-a"
35 $requestedMbps = 7500
36 $Conn2         = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy...
37 # ----- Attributes for connection "3"
38 $connID        = 3
39 $connName      = "Prod-NetworkSet-1"
40 $connType      = "Ethernet"
41 $netName       = "Prod"
42 $ThisNetwork   = Get-HPOVNetworkSet -Name $netName
43 $portID        = "Mezz 3:1-c"
44 $requestedMbps = 2500
45 $Conn3         = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy...
46 # ----- Attributes for connection "4"
```

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

```
Disconnect-HPOVMgmt
```

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



```

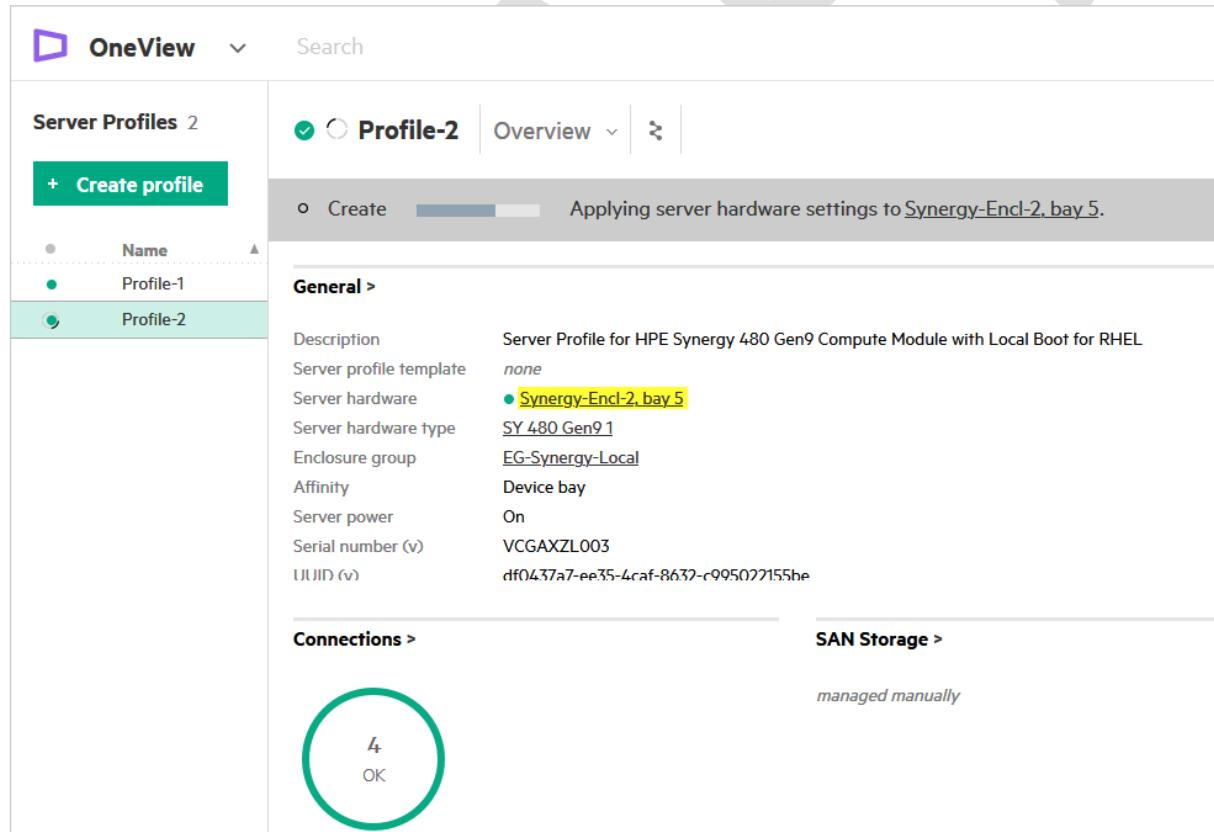
1 # ----- Attributes for ServerProfile "Profile-1"
2 $name          = "Profile-2"
3 $server        = Get-HPOVServer -Name "Synergy-Encl-3, bay 11"
4 $affinity      = "Bay"
5 # ----- Attributes for connection "1"
6 $connID        = 1
7 $connType      = "Ethernet"
8 $netName       = "ESX Mgmt"
9 $ThisNetwork   = 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

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 >

managed manually

4 OK

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 7500Mbps (7.5Gps) requested bandwidth that was defined in our code:

The screenshot shows the HPE OneView interface. On the left, under 'Server Profiles', 'Profile-2' is selected (marked with red circle 1). In the center, under 'Profile-2', the 'Connections' tab is selected (marked with red circle 2). A list of connections is shown, with 'Mgmt-1' expanded (marked with red circle 3). Under 'Mgmt-1', the 'Requested bandwidth' and 'Allocated bandwidth' fields are highlighted with a red box and marked with red circle 4.

ID	Name	Network	Port	Boot
1	Mgmt-1	Mgmt VLAN100 Synergy-Encl-1, interconnect 3	Mezzanine 3:1-a	Not bootable
	Interconnect Type	Ethernet		
	MAC address	1E:CA:38:D0:00:0C (v)		
	Requested virtual functions	None		
	Requested bandwidth	7.5 Gb/s		
	Allocated bandwidth	7.5 Gb/s		
	Max bandwidth	20 Gb/s		
	Link aggregation group	None		
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.

- Create a new file in VS Code

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'SYNTERGY DEMONSTRATIONS ON ...'. Inside this folder, under the 'PowerShell' section, a new file is being created, indicated by a 'New File' button. A red box highlights this 'New File' button. The main editor window displays a PowerShell script with several lines of code, starting with \$deviceSlot and ending with Disconnect-HPOVMgmt.

```

$deviceSlot
$controllerMode
$LogicalDisks
$controller1
$controllers
# -----
$manageboot
$biosBootMode
# -----
$bootOrder
# -----
New-HPOVServerProfile
Disconnect-HPOVMgmt

```

- Name it **Server_FW_Report.ps1**

The screenshot shows the Visual Studio Code interface after renaming the file. The 'Server_FW_Report.ps1' file is now listed in the 'OPEN EDITORS' section of the Explorer sidebar. The main editor window is empty, showing only the number 1.

- Copy/paste the following script content in the new VS Code file:

```
#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-HPOVMgmt -appliance $IP -Credential $credentials

$servers = Get-HPOVServer

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

foreach ($server in $servers) {

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

    $name = (Get-HPOVServer -name $server.name ).name
    $romVersion = (Get-HPOVServer -name $server.name ).romVersion

    foreach ($component in $components) {

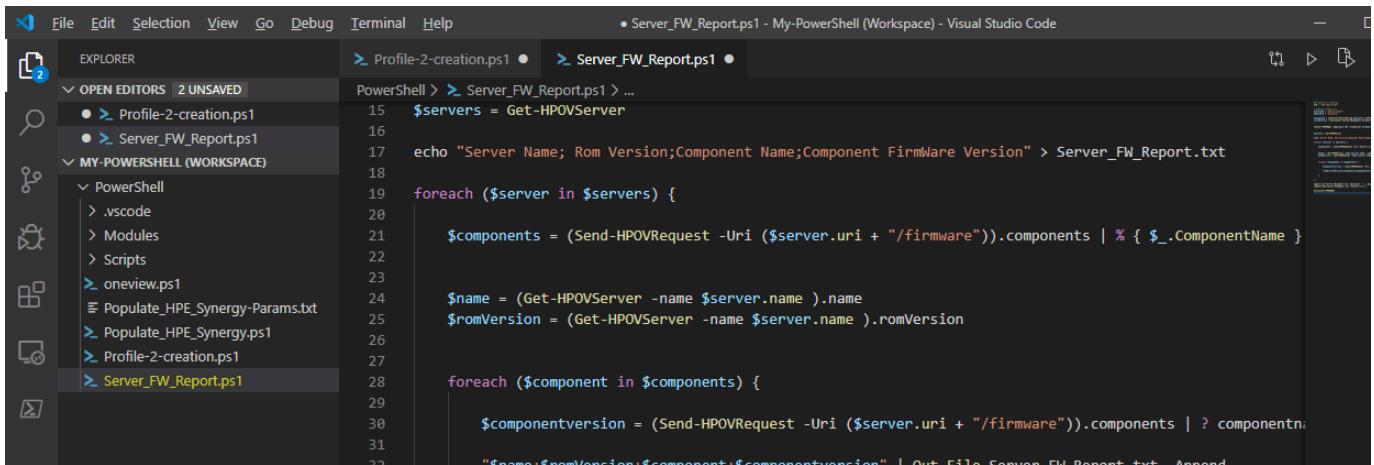
        $componentversion = (Send-HPOVRequest -Uri ($server.uri + "/firmware")).components | ? componentname -eq $component | select componentVersion | % { $_.componentVersion }

        "$name;$romVersion;$component;$componentversion" | Out-File Server_FW_Report.txt -Append
    }
}

import-csv Server_FW_Report.txt -delimiter ";" | export-csv Server_FW_Report.csv -NoTypeInformation
remove-item Server_FW_Report.txt -Confirm:$false

Disconnect-HPOVMgmt
```

- Press **CTRL + S** to save the script

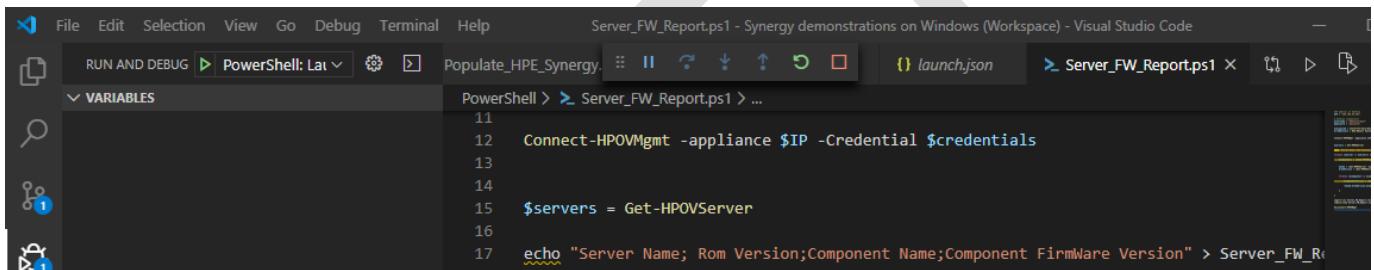


The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Explorer:** Shows two open editors: "Profile-2-creation.ps1" and "Server_FW_Report.ps1".
- Terminal:** PowerShell > Server_FW_Report.ps1 > ...
- Code Editor:** Displays the content of "Server_FW_Report.ps1".

```
15 $servers = Get-HPOVServer
16
17 echo "Server Name; Rom Version;Component Name;Component Firmware Version" > Server_FW_Report.txt
18
19 foreach ($server in $servers) {
20
21     $components = (Send-HPOVRequest -Uri ($server.uri + "/firmware")).components | % { $_.ComponentName }
22
23
24     $name = (Get-HPOVServer -name $server.name).name
25     $romVersion = (Get-HPOVServer -name $server.name).romVersion
26
27
28     foreach ($component in $components) {
29
30         $componentversion = (Send-HPOVRequest -Uri ($server.uri + "/firmware")).components | ? componentname
31
32         "$name:$romVersion:$component:$componentversion" | Out-File Server_FW_Report.txt -Append
```

- Then we can run the script by pressing **F5**



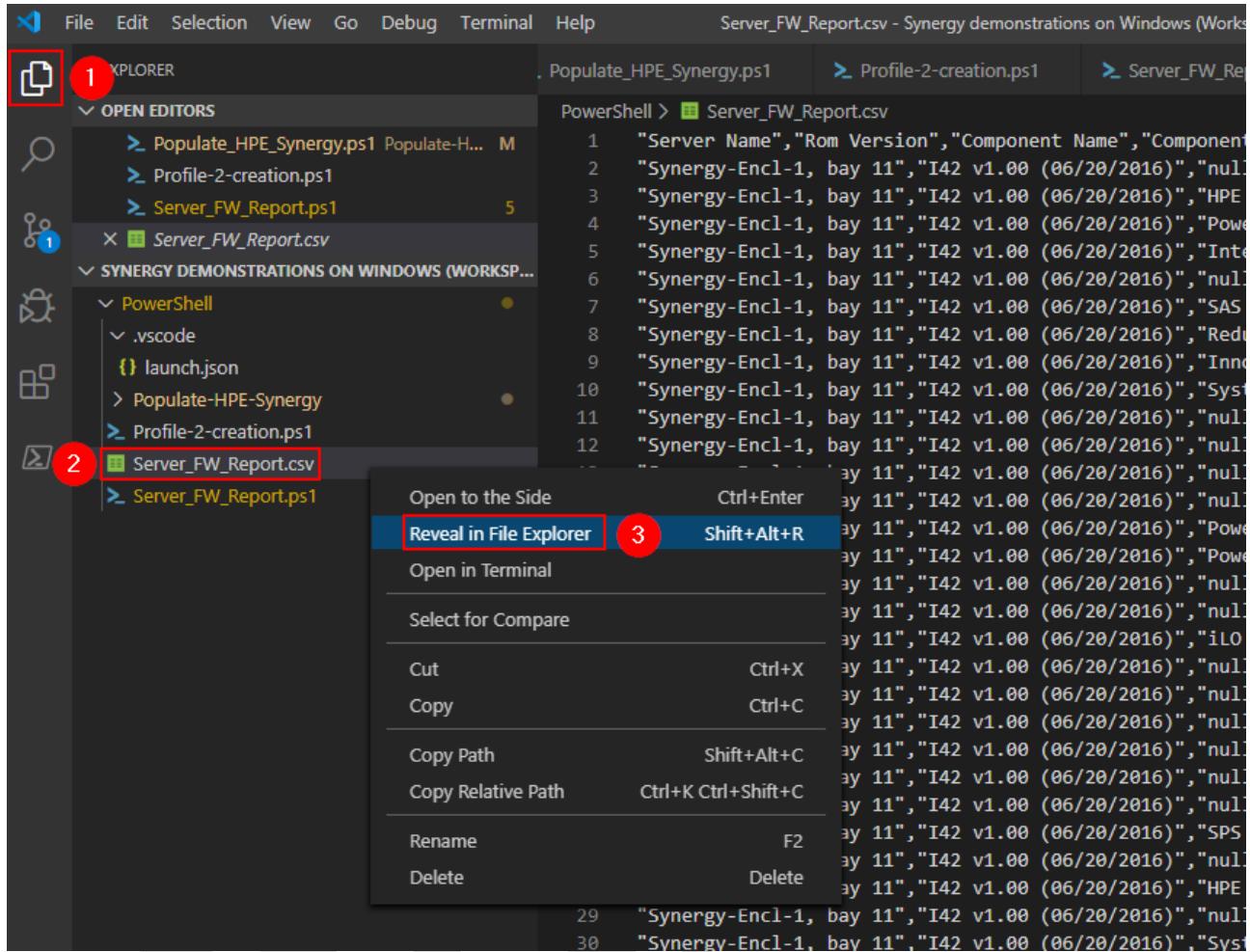
The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Run and Debug:** PowerShell: Launch
- Variables:** Shows the variable \$IP set to 10.0.1.10.
- Terminal:** PowerShell > Server_FW_Report.ps1 > ...
- Code Editor:** Displays the content of "Server_FW_Report.ps1".

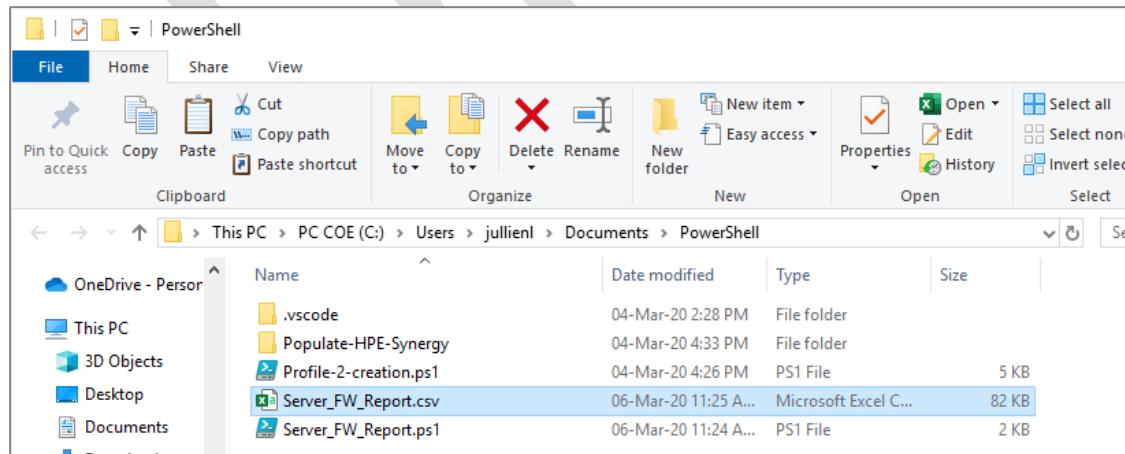
```
11
12 Connect-HPOVMgmt -appliance $IP -Credential $credentials
13
14
15 $servers = Get-HPOVServer
16
17 echo "Server Name; Rom Version;Component Name;Component Firmware Version" > Server_FW_Report.txt
```

The script generates a CSV file in your working folder

- Go back to the **Explorer**, right-click on the new generated CSV file and select **Reveal in File Explorer**



This will open the Windows explorer



- You can then 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:

A	B	C
34	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)
35	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)
36	Synergy-Encl-1, bay 11	null
37	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)
38	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
39	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
40	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
41	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
42	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
43	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
44	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
45	Synergy-Encl-1, bay 3	HPE Synergy 3820C 10/20Gb Converged Network Adapter
46	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
47	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014
48	Synergy-Encl-1, bay 3	iLO
49	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
50	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
51	Synergy-Encl-1, bay 4	Power Management Controller FW Bootloader
52	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
53	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
54	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
55	Synergy-Encl-1, bay 4	HPE Synergy 3820C 10/20Gb Converged Network Adapter
56	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
57	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
58	Synergy-Encl-1, bay 4	HP ProLiant System ROM - Backup
59	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014
60	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
61	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
62	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
63	Synergy-Encl-1, bay 5	HP ProLiant System ROM - Backup
64	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
65	Synergy-Encl-1, bay 5	Power Management Controller FW Bootloader
66	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
67	Synergy-Encl-1, bay 5	HP ProLiant System ROM
68	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
69	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014
		Intelligent Provisioning
		System Programmable Logic Device
		Power Management Controller Firmware
		Power Management Controller FW Bootloader
		Intelligent Platform Abstraction Data
		Server Platform Services (SPS) Firmware
		Intelligent Provisioning
		iLO
		Intelligent Platform Abstraction Data
		Power Management Controller FW Bootloader
		HP ProLiant System ROM
		Intelligent Platform Abstraction Data
		Power Management Controller FW Bootloader
		HP ProLiant System ROM
		System Programmable Logic Device
		Server Platform Services (SPS) Firmware
		Intelligent Platform Abstraction Data
		Power Management Controller Firmware

Note: This report contains 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 Logical Enclosure is available)

- To easily create the appropriate CSV file to generate the new networks, you can copy the following PowerShell script

```
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 $_ }
```

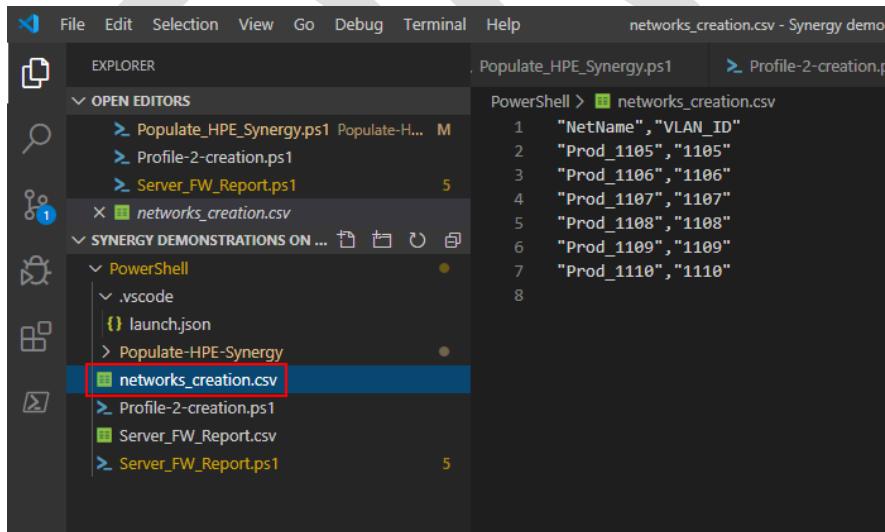
- And paste it in the PowerShell Integrated console:



The screenshot shows the VS Code interface with the PowerShell integrated terminal tab selected. The terminal window displays the PowerShell command to create a CSV file named 'networks_creation.csv' with specific values for 'NetName' and 'VLAN_ID'. The command uses the 'Add-Content' cmdlet and loops through a list of network names and IDs.

```
PS C:\VS Code projects\PowerShell> Add-Content networks_creation.csv -value '"NetName","VLAN_ID"  
PS C:\VS Code projects\PowerShell> $networks = @('Prod_1105","1105','Prod_1106","1106','Prod_1107","1107','Prod_1108","1108','Prod_1109","1109'  
PS C:\VS Code projects\PowerShell> $networks | foreach { Add-Content -Path networks_creation.csv -Value $_ }  
PS C:\VS Code projects\PowerShell> [REDACTED]
```

- This script creates a **networks_creation.csv** in the working directory

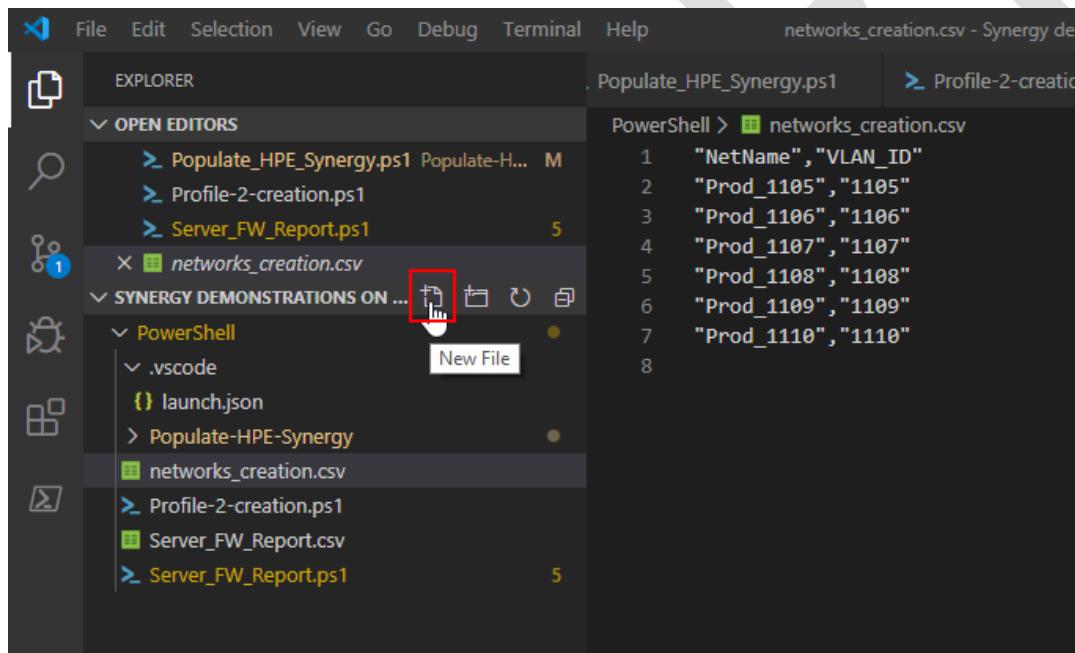


In Excel, it is built as follows:

	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			

This is the spreadsheet that we will use to autogenerate six Ethernet Networks **Prod_1105, Prod_1106, ... Prod_1110** in our Synergy environment.

- Now create a new file in VS Code by pressing **New File**



- Name it **Add_Networks_from_CSV.ps1**
- Copy/paste the following code in this new file:

```
$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-HPOVMgmt -Hostname $IP -Credential $credentials | Out-Null

$data = (Import-Csv $csvfile)

# Creating Networks and adding them to the LIG uplink Set

$LIG = Get-HPOVLogicalInterconnectgroup -Name $LIGname

if (!((($LIG | Measure-Object).Count -eq 1))) { Write-Host "Failed to filter down to one LIG" -ForegroundColor Red | Break }

ForEach ($VLAN In $data) {
    New-HPOVNetwork -Name $VLAN.NetName -Type Ethernet -VLANId $VLAN.VLAN_ID -SmartLink $True | out-Null
    Write-host "`nCreating Network: " -NoNewline
    Write-host -f Cyan $($VLAN.netName) -NoNewline

    (($LIG.uplinkSets | where-object name -eq $LIG_UplinkSet | Where-Object { $_.ethernetNetworkType -eq "Tagged" }).networkUris) += (Get-HPOVNetwork -
    Name $VLAN.NetName).uri #Add NewNetwork to the networkUris Array
    Write-host "`nAdding Network: " -NoNewline
    Write-host -f Cyan $($VLAN.netName) -NoNewline
    Write-host " to Uplink Set: " -NoNewline
    Write-host -f Cyan $LIG_UplinkSet

}

try {
    Set-HPOVResource $LIG -ErrorAction Stop | Wait-HPOVTaskComplete #| Out-Null
}
catch {
    Write-Output $_ #.Exception
}

# Updating LI from LIG

$LI = ((Get-HPOVLogicalInterconnect) | where-object logicalInterconnectGroupUri -eq $LIG.uri)

do {
    $Interconnectstate = (((Get-HPOVInterconnect) | where-object productname -match "Virtual Connect") | where-object logicalInterconnectUri -EQ $LI.uri).state

    if ($Interconnectstate -notcontains "Configured") {

        Write-
host "`nWaiting for the running Interconnect configuration task to finish, please wait...`n"
    }
}

until ($Interconnectstate -notcontains "Adding" -and $Interconnectstate -notcontains "Imported" -and $Interconnectstate -notcontains "Configuring")

Write-host "`nUpdating all Logical Interconnects from the Logical Interconnect Group: " -NoNewline
Write-host -f Cyan $LIG.name
Write-host "`nPlease wait..."

```



```
try {
    Get-HPOVLogicalInterconnect -Name $LI.name | Update-HPOVLogicalInterconnect -confirm:$false -
ErrorAction Stop | Wait-HPOVTaskComplete | Out-Null
}
catch {
    Write-Output $_ #.Exception
}

# Adding Network to Network Set

ForEach ($VLAN In $data) {

    Write-host "`nAdding Network: " -NoNewline
    Write-host -f Cyan ($VLAN.netName) -NoNewline
    Write-host " to NetworkSet: " -NoNewline
    Write-host -f Cyan $networksetname

    $VLANuri = (Get-HPOVNetwork -Name $VLAN.NetName).uri
    $networkset = Get-HPOVNetworkSet -Name $networksetname

    $networkset.networkUris += (Get-HPOVNetwork -Name $VLAN.NetName).uri

    try {
        Set-HPOVNetworkSet $networkset -ErrorAction Stop | Wait-HPOVTaskComplete | Out-Null
    }
    catch {
        Write-Output $_
    }

    if ( (Get-HPOVNetworkSet -Name $NetworkSetname).networkUris -contains $VLANuri) {
        Write-host "`nThe network VLAN ID: " -NoNewline
        Write-host -f Cyan $VLAN.NetName -NoNewline
        Write-
host " has been added successfully to all Server Profiles that are using the Network Set: " -NoNewline
        Write-host -f Cyan $networksetname
    }
    else {
        Write-
Warning "`nThe network VLAN ID: $($VLAN.VLAN_ID) has NOT been added successfully, check the status of y
our Logical Interconnect resource`n"
    }
}

$ConnectedSessions | Disconnect-HPOVMgmt | Out-Null
```

- Save the file by pressing **CTRL + S** then press **F5** to run the script



- 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
[oooooooooooo
[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 Status
Deployment	1500	Ethernet	Completed
ESX Mgmt	1131	Ethernet	Pending
ESX vMotion	1132	Ethernet	Pending
Mgmt	100	Ethernet	Pending
Prod_1101	1101	Ethernet	Pending
Prod_1102	1102	Ethernet	Pending
Prod_1103	1103	Ethernet	Pending
Prod_1104	1104	Ethernet	Pending
Prod_1105	1105	Ethernet	Completed
Prod_1106	1106	Ethernet	Pending
Prod_1107	1107	Ethernet	Pending
Prod_1108	1108	Ethernet	Pending
Prod_1109	1109	Ethernet	Pending
Prod_1110	1110	Ethernet	Pending
SAN A FC		FC	Pending

- 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 center, the 'Logical Interconnect' details for 'LE-Synergy-Local-LIG-FlexFabric' show five categories: Internal (no networks), Mgmt (1 network, 2 uplink ports), Prod (10 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 view of 'Synergy-Encl-1' shows a 4x4 port grid. 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' section is displayed for 'Prod'. It shows 'Mgmt' and 'Prod' uplink sets. The 'Prod' uplink set (highlighted with a red box) has 3 entries. The 'Prod' entry (highlighted with a red box) shows connection mode as 'Automatic', LACP timer as 'Short (1s)', and LACP load balancing as 'Source & Destination MAC Address'. It also lists LACP failover trigger, LACP distribute uplink ports, and Native network. Below this, the 'Networks (10)' section lists ten network pairs, with the second pair ('Prod_1106_1106') highlighted in yellow. Other sections include 'Network sets' (No network sets) and 'Uplinks'.

- 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" and a log entry from "Administrator" at "1/29/20 9:07:11 am". Below this, the "General" section displays the following information:

Preferred bandwidth	2.5 Gb/s
Maximum bandwidth	20 Gb/s
Type	Regular
Used by	2 server profiles

Below the general section is a "Networks" section showing a grid of network names:

Prod_1101	1101	Prod_1103	1103	Prod_1105	1105	Prod_1107	1107	Prod_1109	1109
Prod_1102	1102	Prod_1104	1104	Prod_1106	1106	Prod_1108	1108	Prod_1110	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

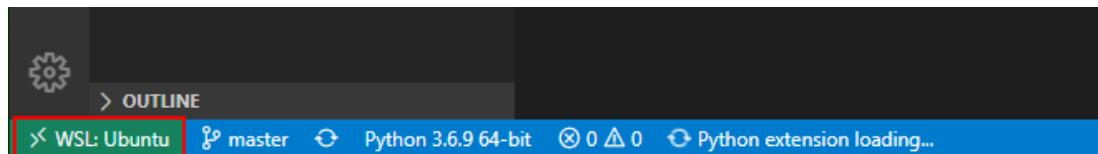
This concludes PowerShell – Scenario 3

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.

Prerequisites:

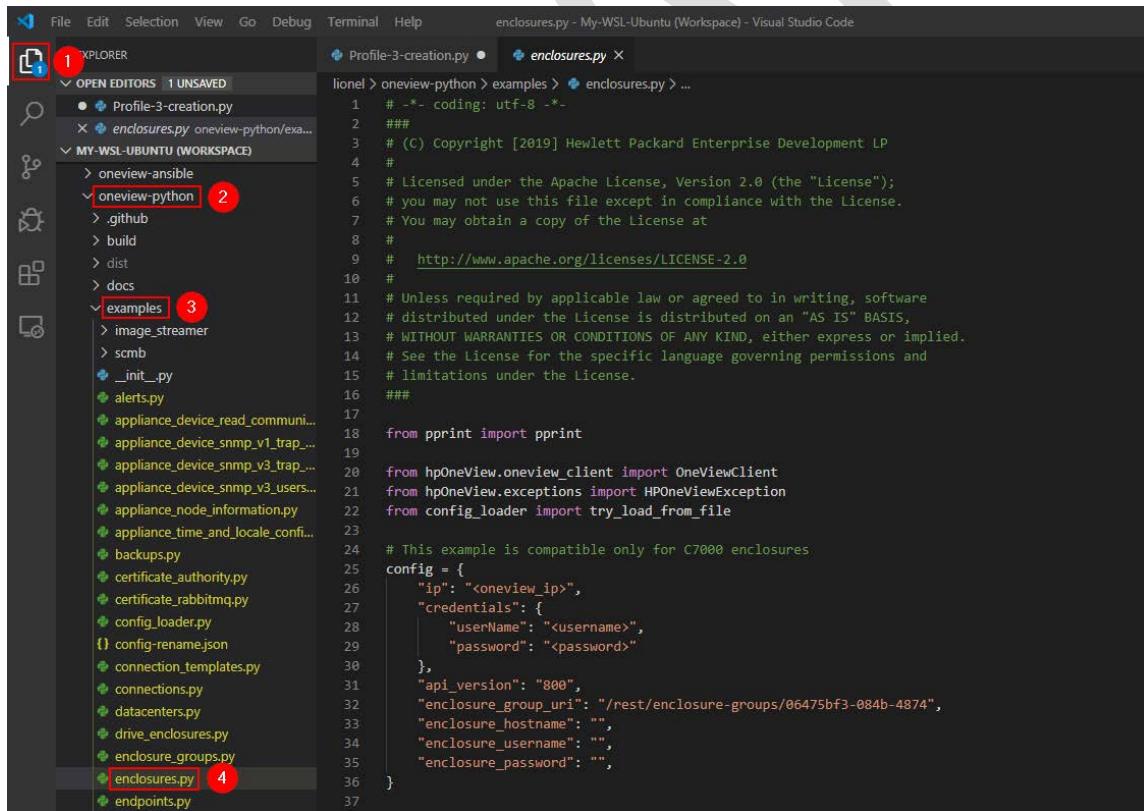
- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



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

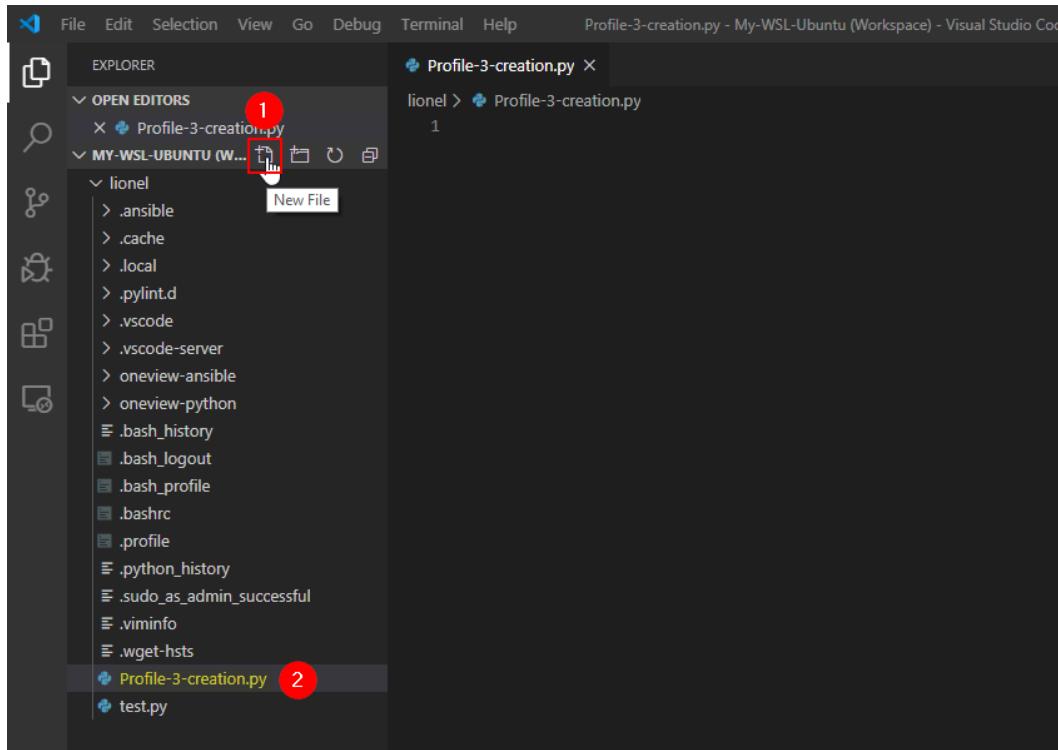
Before creating a server profile, you can show the list of examples provided by the *OneView-Python* library.

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



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

- Now close the *oneview-python* folder and create in your Ubuntu user home directory a new file, name it **Profile-3-creation.py**



- Then add the following lines to your code:

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

config = {
    "ip": "192.168.56.101",
    "api_version": 1200,
    "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)

server_profiles = oneview_client.server_profiles

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

A few explanations:

- 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": 1200,
    "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 7' from `server_profile_templates`:

```
server = server_hardwares.get_by_name('Synergy-Encl-3, bay 7')
```

- 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'
```

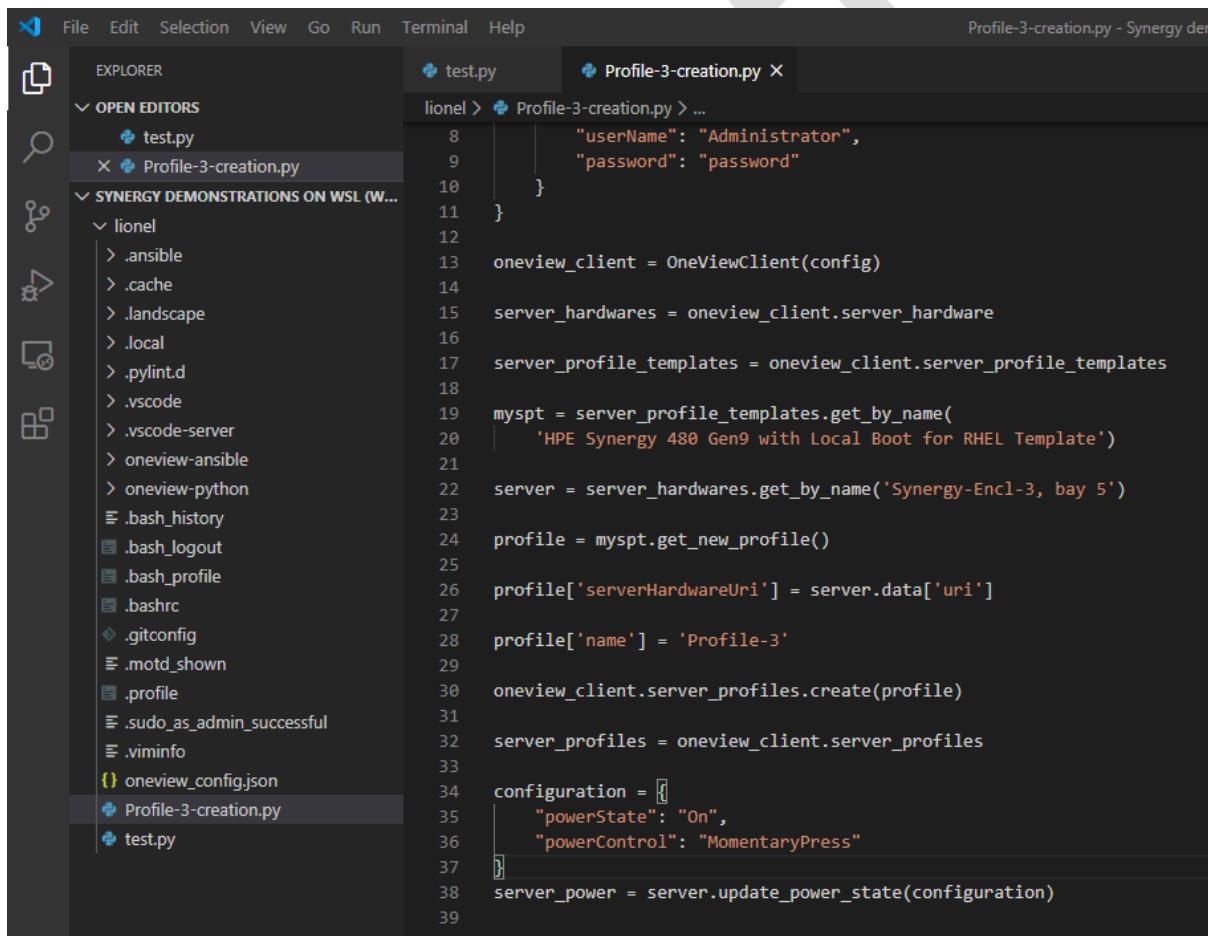
- Create a new profile from our Server Profile Template:
`oneview_client.server_profiles.create(profile)`

- Turn the server on:

```
configuration = {
    "powerState": "On",
    "powerControl": "MomentaryPress"
}

server_power = server.update_power_state(configuration)
```

- Save now the script, press **CTRL + S**.



The screenshot shows a code editor window with a dark theme. On the left is an Explorer sidebar showing project files: test.py, Profile-3-creation.py, .ansible, .cache, .landscape, .local, .pylint.d, .vscode, .vscode-server, oneview-ansible, oneview-python, .bash_history, .bash_logout, .bash_profile, .bashrc, .gitconfig, .motd_shown, .profile, .sudo_as_admin_successful, .viminfo, oneview_config.json, Profile-3-creation.py, and test.py. The main editor area displays the following Python script:

```
profile['name'] = 'Profile-3'

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)

server_profiles = oneview_client.server_profiles

configuration = [
    "powerState": "On",
    "powerControl": "MomentaryPress"
]

server_power = server.update_power_state(configuration)
```

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

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

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** Profile-3-creation.py - My-WSL-Ubuntu (Workspace) - Visual Studio Code.
- Left Sidebar:** RUN AND DEBUG (highlighted), Python: Current, Variables, Search, Schemas, Diagnostics, Problems, and a file icon.
- Code Editor:** The current file is Profile-3-creation.py, containing Python code to initialize an OneViewClient. The code imports OneViewClient and pprint, defines a config dictionary with IP, API version, and credentials, and initializes the client and hardware profiles.
- Bottom Bar:** OUTPUT, TERMINAL, DEBUG CONSOLE, PROBLEMS, and a Python Debug Console tab.
- Terminal:** The terminal shows the command /usr/bin/python3 being run followed by the path to the current script.

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

The screenshot shows the HPE OneView interface for managing server profiles. On the left, a sidebar lists 'Server Profiles 3' with a '+ Create profile' button. The main area displays 'Profile-3' in edit mode, with a progress bar indicating 'Applying server hardware settings to Synergy-Encl-3, bay 5.' The 'General' tab is selected, showing detailed configuration for the server profile, including:

Name	Value
Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	HPE Synergy 480 Gen9 with Local Boot for RHEL Template
Server hardware	Synergy-Encl-3, bay 5
Server hardware type	SY 480 Gen9.1
Enclosure group	EG-Synergy-Local
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

The 'OS Deployment' section at the bottom indicates 'not set'.

- Once created, the server is turned on:

The screenshot shows the HPE OneView interface. On the left, there's a sidebar titled "Server Profiles 3" with a green button "+ Create profile". Below it is a list of profiles: Profile-1, Profile-2, and Profile-3, with Profile-3 selected and highlighted in green. The main panel shows a summary for "Profile-3" under the "General" tab. It includes a progress bar indicating "Create Completed 4m6s". The "General" section displays the following details:

Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	HPE Synergy 480 Gen9 with Local Boot for RHEL Template
Server hardware	Synergy-Encl-3_bay5
Server hardware type	SY 480 Gen9 1
Enclosure group	EG-Synergy-Local
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

This concludes Python – Scenario 1



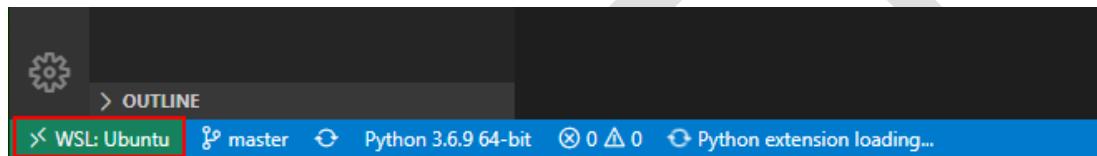
Python – Scenario 2 – Creating a report

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

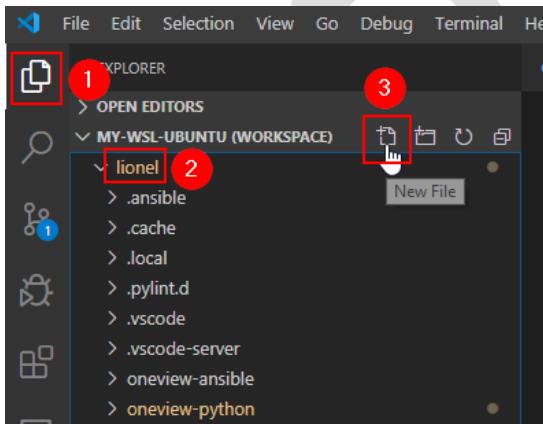
Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



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

- Create a new file in VS Code in your user home directory, name it **Server_FW_Report.py**



- Then add the following lines to your script:

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

config = {
    "ip": "192.168.56.101",
    "api_version": 1200,
    "credentials": {
        "userName": "Administrator",
        "password": "password"
    }
}

oneview_client = OneViewClient(config)
server_hardwares = oneview_client.server_hardware
server_hardware_gen9 = server_hardwares.get_by("shortModel", "SY 480 Gen9")

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

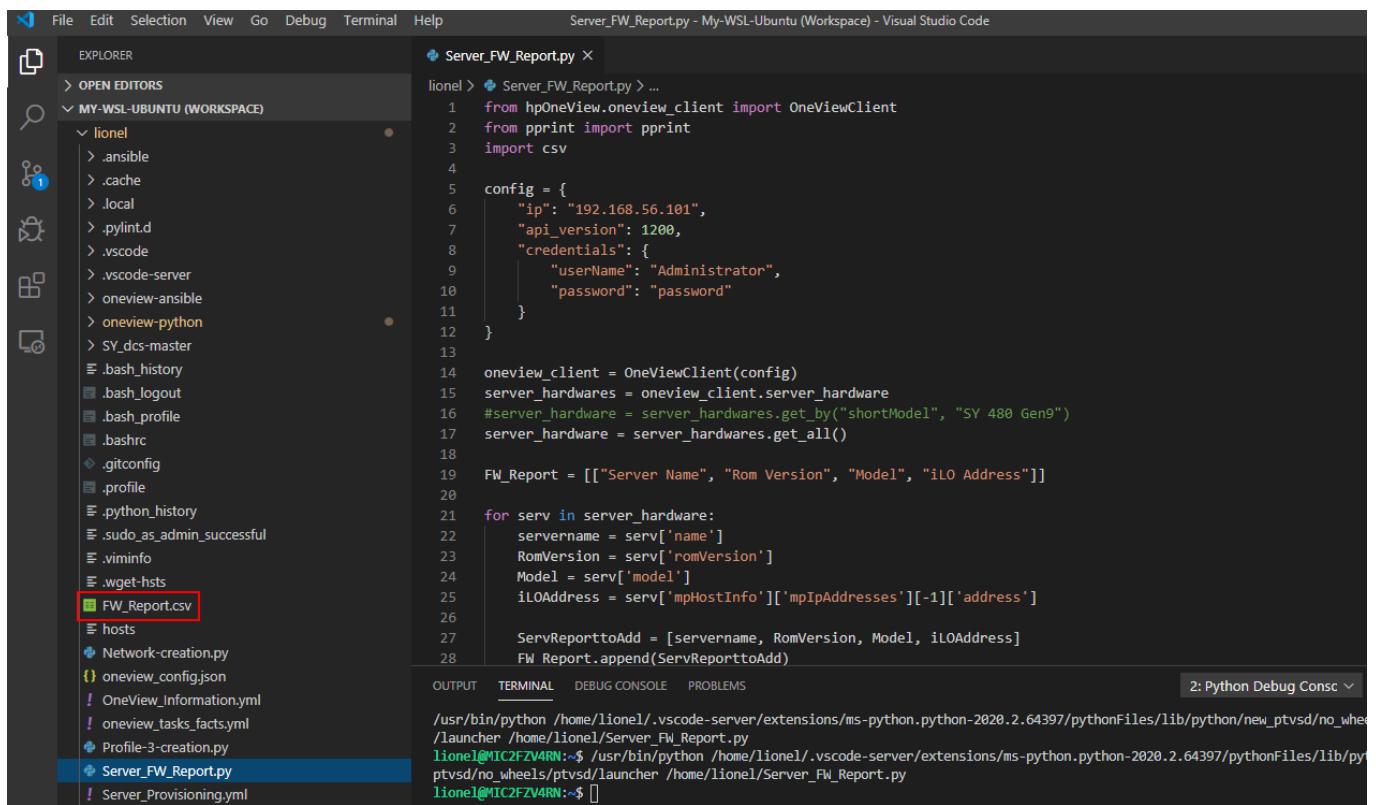
for serv in server_hardware_gen9:
    servername = serv['name']
    RomVersion = serv['romVersion']
    Model = serv['model']
    iLOAddress = serv['mpHostInfo']['mpIpAddresses'][-1]['address']

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

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

- Then save the script by pressing **CTRL + S**, then **F5** to run the script.

- Once run, you should find the **FW_Report.csv** file in your user home directory:



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders under 'MY-WSL-UBUNTU (WORKSPACE)'. A red box highlights the 'FW_Report.csv' file. The main editor area displays the Python script 'Server_FW_Report.py'. The terminal at the bottom shows the command 'python Server_FW_Report.py' being run, and the output shows the CSV file being generated.

```

File Edit Selection View Go Debug Terminal Help
Server_FW_Report.py - My-WSL-Ubuntu (Workspace) - Visual Studio Code

EXPLORER
> OPEN EDITORS
MY-WSL-UBUNTU (WORKSPACE)
lionel
  > .ansible
  > .cache
  > .local
  > .pylint.d
  > .vscode
  > .vscode-server
  > oneview-ansible
  > oneview-python
  > SY_dcs-master
  > .bash_history
  > .bash_logout
  > .bash_profile
  > .bashrc
  > .gitconfig
  > .profile
  > .python_history
  > .sudo_as_admin_successful
  > .viminfo
  > .wget-hsts
  > FW_Report.csv
  > hosts
  & Network-creation.py
  { onewview_config.json
  ! OneView_Information.yml
  ! onewview_tasks_facts.yml
  & Profile-3-creation.py
  & Server_FW_Report.py
  ! Server_Provisioning.yml
  ! Server_Unprovisioning
  ! test.yml
  & testSP.py

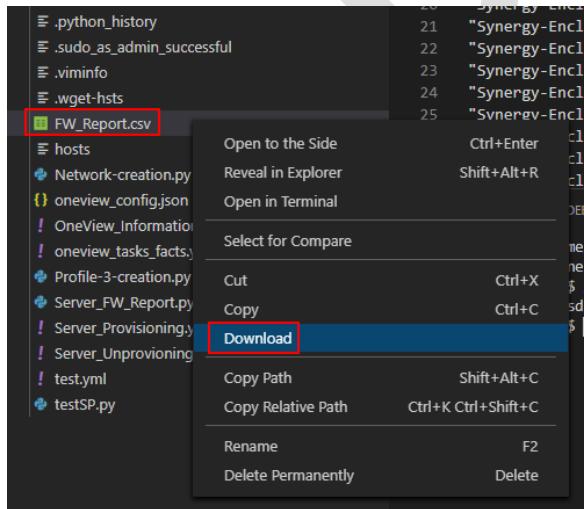
Server_FW_Report.py
Server_Provisioning.yml

IONEL > Server_FW_Report.py > ...
lionel > Server_FW_Report.py > ...
1   from hpOneView.oneview_client import OneViewClient
2   from pprint import pprint
3   import csv
4
5   config = {
6       "ip": "192.168.56.101",
7       "api_version": 1200,
8       "credentials": {
9           "userName": "Administrator",
10          "password": "password"
11      }
12  }
13
14  oneview_client = OneViewClient(config)
15  server_hardwares = oneview_client.server_hardware
16  #server_hardware = server_hardwares.get_by("shortModel", "SY 480 Gen9")
17  server_hardware = server_hardwares.get_all()
18
19  FW_Report = [[ "Server Name", "Rom Version", "Model", "iLO Address"]]
20
21  for serv in server_hardware:
22      servername = serv[ 'name' ]
23      RomVersion = serv[ 'romVersion' ]
24      Model = serv[ 'model' ]
25      iLOAddress = serv[ 'mpHostInfo' ][ 'mpIpAddresses' ][ -1 ][ 'address' ]
26
27      ServReportToAdd = [servername, RomVersion, Model, iLOAddress]
28      FW_Report.append(ServReportToAdd)

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS
2: Python Debug Consc <
/usr/bin/python /home/lionel/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lionel/Server_FW_Report.py
lionel@MIC2FZV4RN:~$ /usr/bin/python /home/lionel/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lionel/Server_FW_Report.py
lionel@MIC2FZV4RN:~$ []

```

- 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	

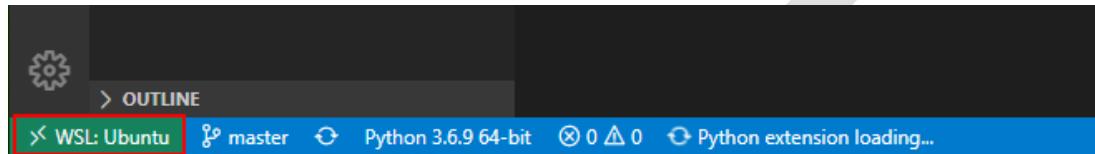
This concludes Python – Scenario 2

Python – Scenario 3 - Accelerating a configuration change

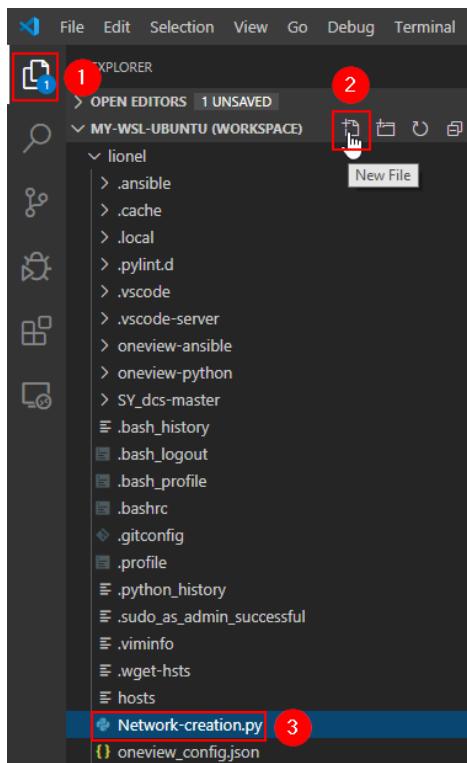
In this scenario, we are going to use Python to create a new network in OneView (Name: *RHEL Prod*, VLANID: 50), then add this network to the uplink set *US-Prod* and to the network set *Prod* so that any of the existing Server Profiles using the network set *Prod* will be automatically connected to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

Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



- **Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)
- In VS Code, go to the **Explorer** to create in your user home directory a new file, name it **Network-creation.py**



- Then add the following lines to your script:

```

from hpOneView.oneview_client import OneViewClient
from pprint import pprint
import json

lig_name = "LIG-FlexFabric"
uplinkset_name = "Prod"
networkset_name = "Prod"

config = {
    "ip": "192.168.56.101",
    "api_version": 1200,
    "credentials": {
        "userName": "Administrator",
        "password": "password"
    }
}

oneview_client = OneViewClient(config)

ethernet_networks = oneview_client.ethernet_networks
network_sets = oneview_client.network_sets

# Creating Ethernet network

options_ethernet = {
    "name": "RHEL Prod",
    "vlanId": 50,
    "ethernetNetworkType": "Tagged",
    "purpose": "General",
    "smartLink": False,
    "privateNetwork": False,
    "connectionTemplateUri": None,
}

ethernet_network = ethernet_networks.create(options_ethernet)

print("Created ethernet-networks '%s' successfully.\n  uri = '%s' "
      % ( ethernet_network.data['name'] , ethernet_network.data['uri'] ) )

# Get RHEL Prod network URI & name

ethernet_network_uri = ethernet_network.data['uri']
ethernet_network_name = options_ethernet['name']

logical_interconnect_groups = oneview_client.logical_interconnect_groups

# Get logical interconnect group by name
lig = logical_interconnect_groups.get_by_name(lig_name)
lig_response = lig.data

#help(oneview_client.logical_interconnect_groups)

for uplink in lig_response['uplinkSets']:
    if uplink['name'] == uplinkset_name :
        new_uplinkset_Prod_networkuris = uplink['networkUris'] + [ethernet_network_uri]
        # pprint(uplink['networkUris'])
        # pprint(new_uplinkset_Prod_networkuris)
        uplink['networkUris'] = new_uplinkset_Prod_networkuris

```

```
lig_to_update = lig_response.copy()
#print(json.dumps(lig_to_update['uplinkSets'], indent=4))

# Update a logical interconnect group
print("Updating logical interconnect group '%s' " % (lig_name))
lig.update(lig_to_update)
print("Number of networks configured in the uplink set '%s' is now %s " % ( uplinkset_name ,
str(len(new_uplinkset_Prod_networkuris))    ))

# Updating the LI from the LIG
logical_interconnect_name = "LE-Synergy-Local-LIG-FlexFabric"
logical_interconnects = oneview_client.logical_interconnects
logical_interconnect = logical_interconnects.get_by_name(logical_interconnect_name)

# Return the logical interconnect to a consistent state
print("Return the logical interconnect to a consistent state")
logical_interconnect_updated = logical_interconnect.update_compliance()
print(" Done. The current consistency state is
{consistencyStatus}.".format(**logical_interconnect_updated))

# Adding new network to Network Set
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)

print("Updated network set '%s' successfully! \n" % (networkset_name))
```

- Then save the script by pressing **CTRL + S**, then **F5** to run the script.

- Simultaneously, you can open the OneView web interface to show the script progress, open the **Networks** page to show the new **RHEL Prod** network:

The screenshot shows the OneView web interface with the 'Networks' page selected. A red box highlights the 'RHEL Prod' row in the list, which is also circled with a red number '2'. The details for the 'RHEL Prod' network are displayed in the right panel under the 'General' tab. The 'RHEL Prod' network is listed with the following properties:

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

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 then go to the **Uplink Sets** sections to show the new **RHEL Prod** network in the **US-Prod** uplink set:

The screenshot shows the OneView web interface with the 'Logical Interconnects' page selected. A red box highlights the 'LE-Synergy-Local-LIG-FlexFabric' logical interconnect, which is also circled with a red number '2'. The details for the 'LE-Synergy-Local-LIG-FlexFabric' logical interconnect are displayed in the right panel under the 'Uplink Sets' tab. The logical interconnect is inconsistent with the logical interconnect group 'LIG-FlexE...'.

Name
LE-Synergy-Local-LIG-FC-1
LE-Synergy-Local-LIG-FC-2
LE-Synergy-Local-LIG-FC-3
LE-Synergy-Local-LIG-FlexFabric
LE-Synergy-Local-LIG-SAS-1
LE-Synergy-Local-LIG-SAS-2
LE-Synergy-Local-LIG-SAS-3

Uplink Sets

Mgmt
Prod

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

Networks (11)

Prod_1101	1101	Prod_1103	1103	Prod_1105	1105	Prod_1107	1107	Prod_1109	1109	RHEL Prod	50
Prod_1102	1102	Prod_1104	1104	Prod_1106	1106	Prod_1108	1108	Prod_1110	1110		

Network sets: 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. In the top left, there's a navigation bar with 'OneView' and a search bar. Below it, a sidebar on the left lists 'Network Sets' (with one item) and a green button '+ Create network set'. The main content area is titled 'Prod' under 'Overview'. It shows a status bar with 'Update Completed' and a timestamp 'Administrator 2/26/20 3:08:25 pm'. A table section titled 'General' contains the following data:

Preferred bandwidth	2.5 Gb/s
Maximum bandwidth	20 Gb/s
Type	Regular
Used by	none

Below this is a 'Networks' section with a table:

Prod 1101	1101	Prod 1102	1102	Prod 1103	1103	Prod 1104	1104	RHEL Prod	50
-----------	------	-----------	------	-----------	------	-----------	------	-----------	----

The 'RHEL Prod' row is highlighted with a yellow background.

- In the console, the following is displayed:

```
lionel@MIC2FZV4RN:~$ env PTVD__LAUNCHER PORT=52062 /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python.python-2020.2.64  
397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lionel/testSP.py  
Created ethernet-networks successfully.  
uri = '/rest/ethernet-networks/f04b7bee-6287-4325-b5e1-b8a23e3f0143'  
The uplink set with name = 'US-Prod' have now the networks:  
['/rest/ethernet-networks/51f797ed-5748-4bd7-bae7-81d827c200c6', '/rest/ethernet-networks/2d9fd879-db86-41c8-b292-9384c8f4de42', '/r  
est/ethernet-networks/f04b7bee-6287-4325-b5e1-b8a23e3f0143', '/rest/ethernet-networks/f8603383-5e3a-4d61-bc0c-29d98317658e', '/rest/e  
thernet-networks/314c5174-129e-4326-b239-0ba3de253b08']  
Updated network set 'Prod' successfully.  
lionel@MIC2FZV4RN:~$
```

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](#)

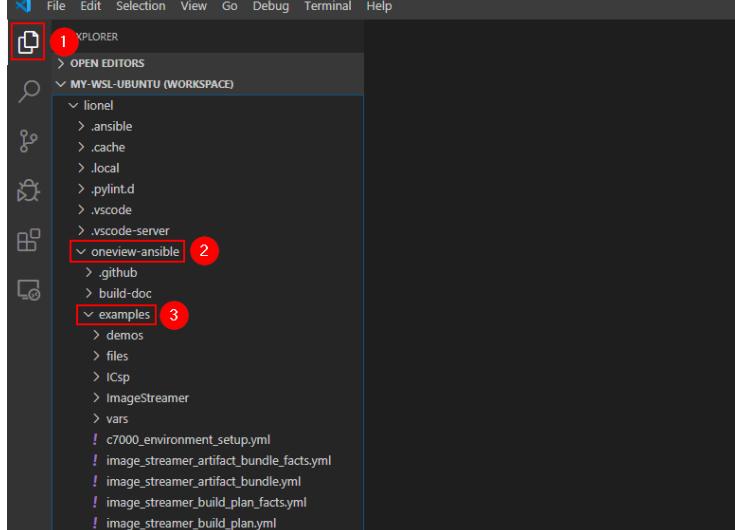
In this scenario, we are going to use Ansible to collect some information from our Synergy Composer DCS appliance.

Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:

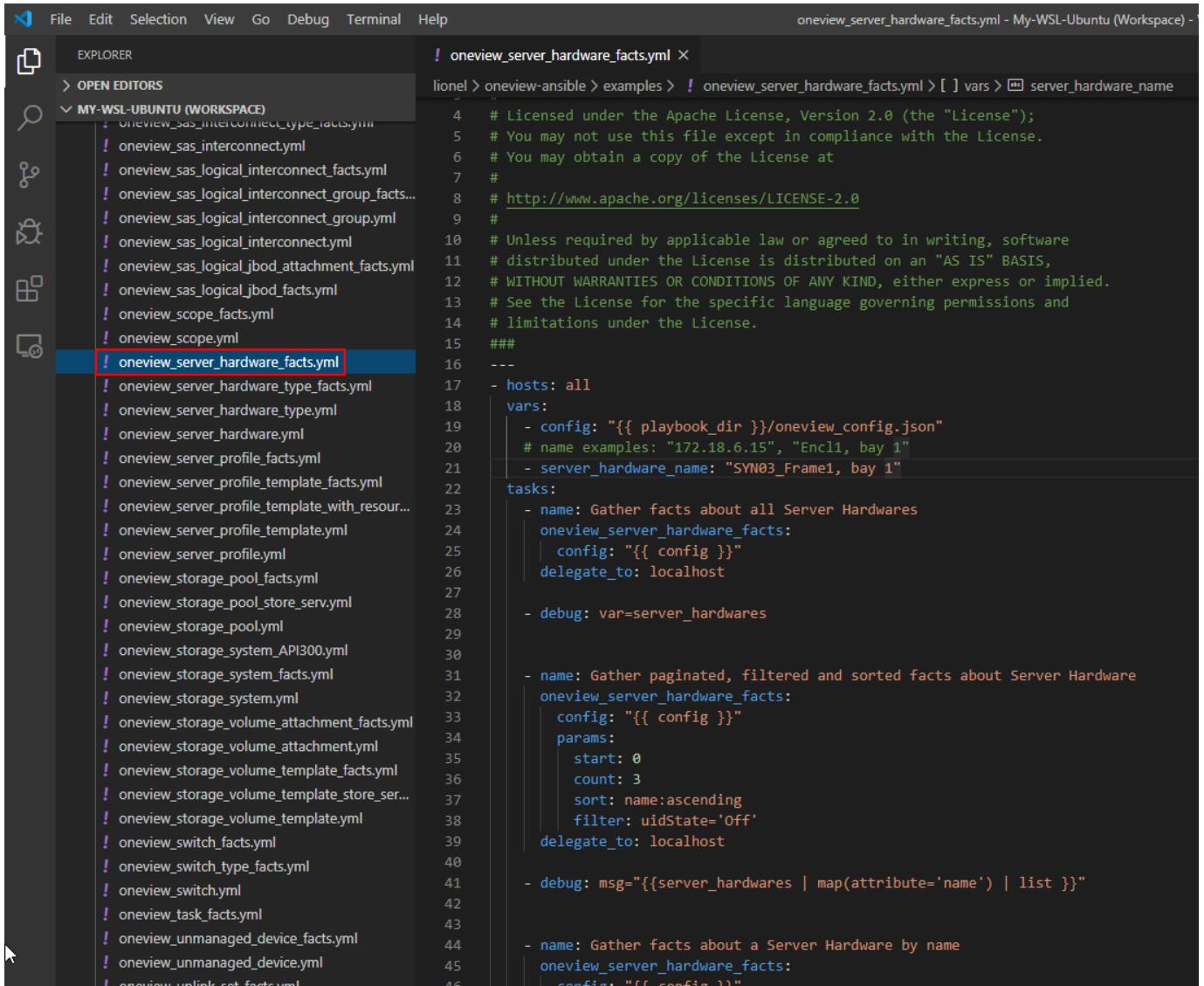


- **Notice:** This scenario can be run with either the initial or final configuration snapshot.
- In Explorer, open the **oneview-ansible** folder and browse the **examples** folder:



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:



```

! 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", "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 }}"

```

- 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
```

The diagram shows four numbered callouts (1, 2, 3, 4) pointing to specific sections of the Ansible playbook code:

- Callout 1:** Points to the first task block, which gathers facts about all server hardwares using the `oneview_server_hardware_facts` module.
- Callout 2:** Points to the second task block, which gathers paginated, filtered, and sorted facts about server hardware using the `oneview_server_hardware_facts` module.
- Callout 3:** Points to the third task block, which gathers facts about a specific server hardware by name using the `oneview_server_hardware_facts` module.
- Callout 4:** Points to the fourth task block, which gathers BIOS facts about a specific server hardware using the `oneview_server_hardware_facts` module.

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/`



A screenshot of a Microsoft Visual Studio Code (VS Code) interface. The title bar shows "File Edit Selection View Go Debug Terminal Help" and the file path "oneview_server_hardware_facts.yml - My-WSL-Ubuntu (Work)". The left sidebar (EXPLORER) shows a file tree under "MY-WSL-UBUNTU (WORKSPACE)": "lionel", ".ansible", ".cache", ".local", ".pylint.d", ".vscode", ".vscode-server", "oneview-ansible" (which contains ".github", "build-doc", "examples", and "library"). The "library" folder is highlighted with a red box. The main editor area displays the content of "oneview_server_hardware_facts.yml". The code is a YAML file with Ansible tasks:

```
lionel > oneview-ansible > examples > oneview_server_hardware_facts.yml > tasks
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:
```

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

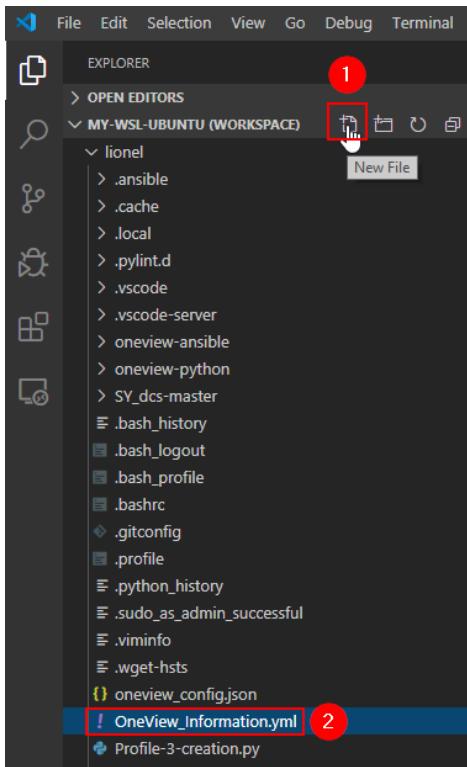
- As we can see, this module returns numerous values: `server_hardwares`, `server_hardware_bios`, `server_hardware_env_config`, etc. These are variables you can use in playbook to run additional tasks.

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 let us create a new file in your user home directory named **OneView_Information.yml**



- Then copy/paste the following content:

```
---
```

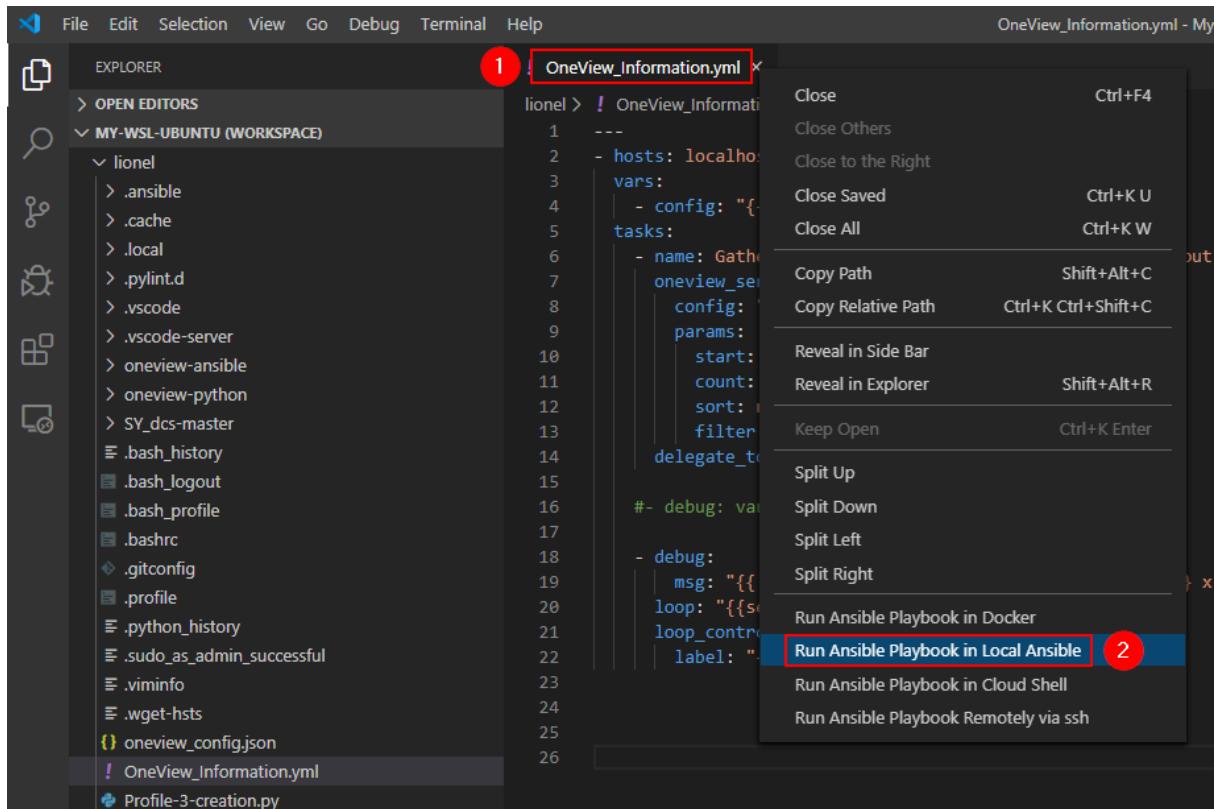
```
- hosts: localhost
  vars:
    - config: "{{ playbook_dir }}/oneview_config.json"
  tasks:
    - name: Gather paginated, filtered and sorted facts about Server Hardware
      oneview_server_hardware_facts:
        config: "{{ config }}"
        params:
          start: 0
          count: 3
          sort: name:ascending
          filter: uidState='Off'
        delegate_to: localhost

      #- debug: var=server_hardwares

      - debug:
          msg: "{{ item.name }} has {{ item.processorCount }} x {{ item.processorType }} processors"
          loop: "{{server_hardwares}}"
          loop_control:
            label: "{{ item.model }}"
```

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

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



Notice that VS Code automatically generates and runs the command `ansible-playbook "/home/<username>/oneview_alerts.yml"` in the terminal window:

The terminal window shows the following output:

```
lionel@MIC2FZV4RN:~$ export VSCODEEXT_USER_AGENT=vscooss.vscode-ansible-0.5.2  
lionel@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/OneView_Information.yml"
```

- The console outputs the following:

```
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Gather paginated, filtered and sorted facts about Server Hardware] ****
ok: [localhost -> localhost]

TASK [debug] ****
ok: [localhost] => (item=Synergy 480 Gen10) => {
    "msg": "Synergy-Encl-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-Encl-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-Encl-1, bay 4 has 4 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}

PLAY RECAP ****
localhost : ok=3    changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
```

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 ansible-tmp-1591538856.968808-12936-59109817527365=\"` echo /home/`/.ansible/tmp/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>

Formatted Server hardware information is displayed through our Ansible debug message task. We are using the `server_hardwares` variable generated by the `oneview_server_hardware_facts` module:

```
- debug:
  msg: "{{ item.name }} has {{ item.processorCount }} x {{ item.processorType }} processors"
  loop: "{{server_hardwares}}"
  loop_control:
    label: "{{ item.model }}"
```

We use the `loop` option to go through the content of `server_hardwares` a dictionary object.

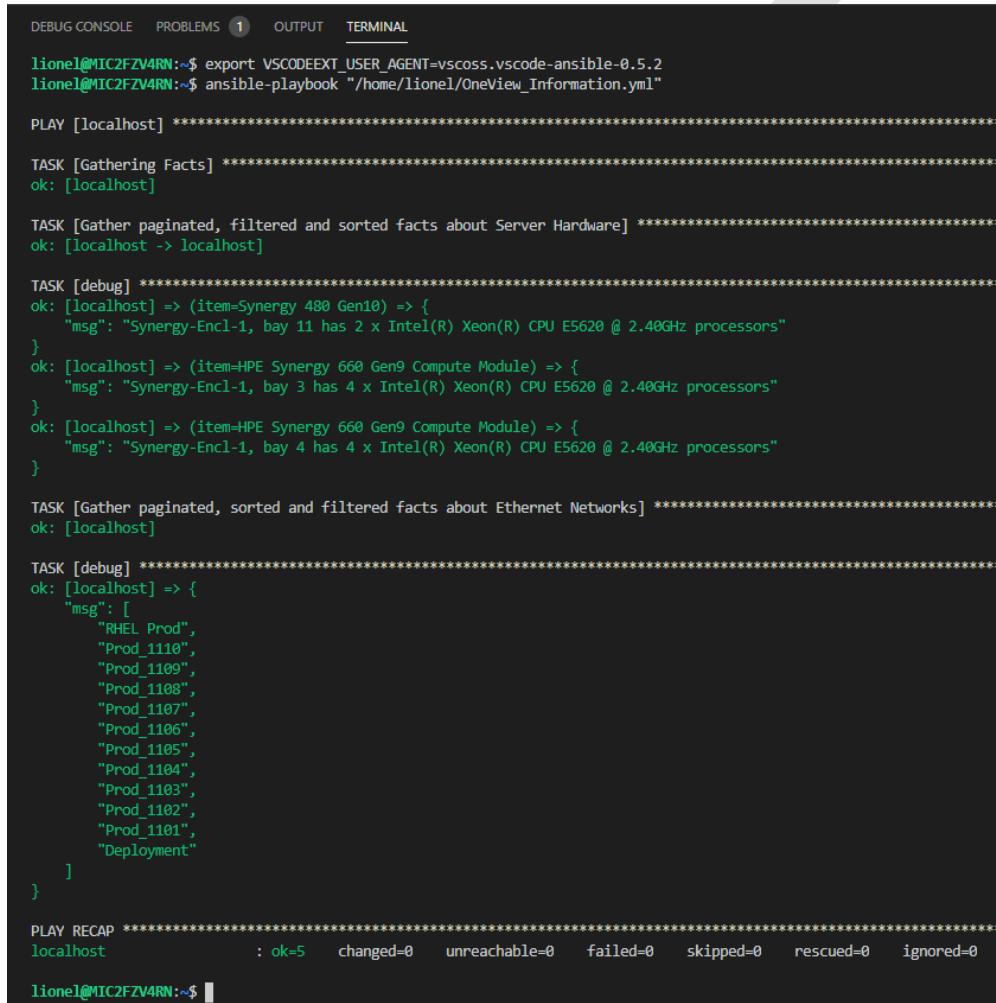
`loop_control` is used to avoid Ansible to display the entire content of the `{{ item }}` variable but instead just the `model`.

- Next, as a second fact example, we can collect some networks available in OneView by adding in the same playbook at the end:

```
- name: Gather paginated, sorted and filtered facts about Ethernet Networks
  oneview_ethernet_network_facts:
    config: "{{ config }}"
    params:
      sort: 'name:descending'
      filter: "purpose=General"

- debug: msg="{{ ethernet_networks | map(attribute='name') | list }}"
```

- Save the file then run it:



```
DEBUG CONSOLE PROBLEMS 1 OUTPUT TERMINAL
lionel@MIC2FZV4RN:~$ export VSCODEEXT_USER_AGENT=vscooss.vscode-ansible-0.5.2
lionel@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/OneView_Information.yml"

PLAY [localhost] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Gather paginated, filtered and sorted facts about Server Hardware] *****
ok: [localhost -> localhost]

TASK [debug] *****
ok: [localhost] => (item=Synergy 480 Gen10) => {
    "msg": "Synergy-Encl-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-Encl-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-Encl-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_110",
        "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:~$
```

Networks with a general purpose are displayed using a descending sort.

We can gather facts about all resources in OneView using parameters found in the module documentation to sort, filter, count, etc.

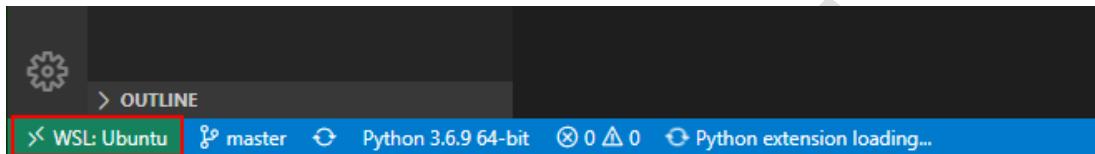
This concludes Ansible – Scenario 1 about collecting facts in OneView

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.

Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



- **Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a 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
```

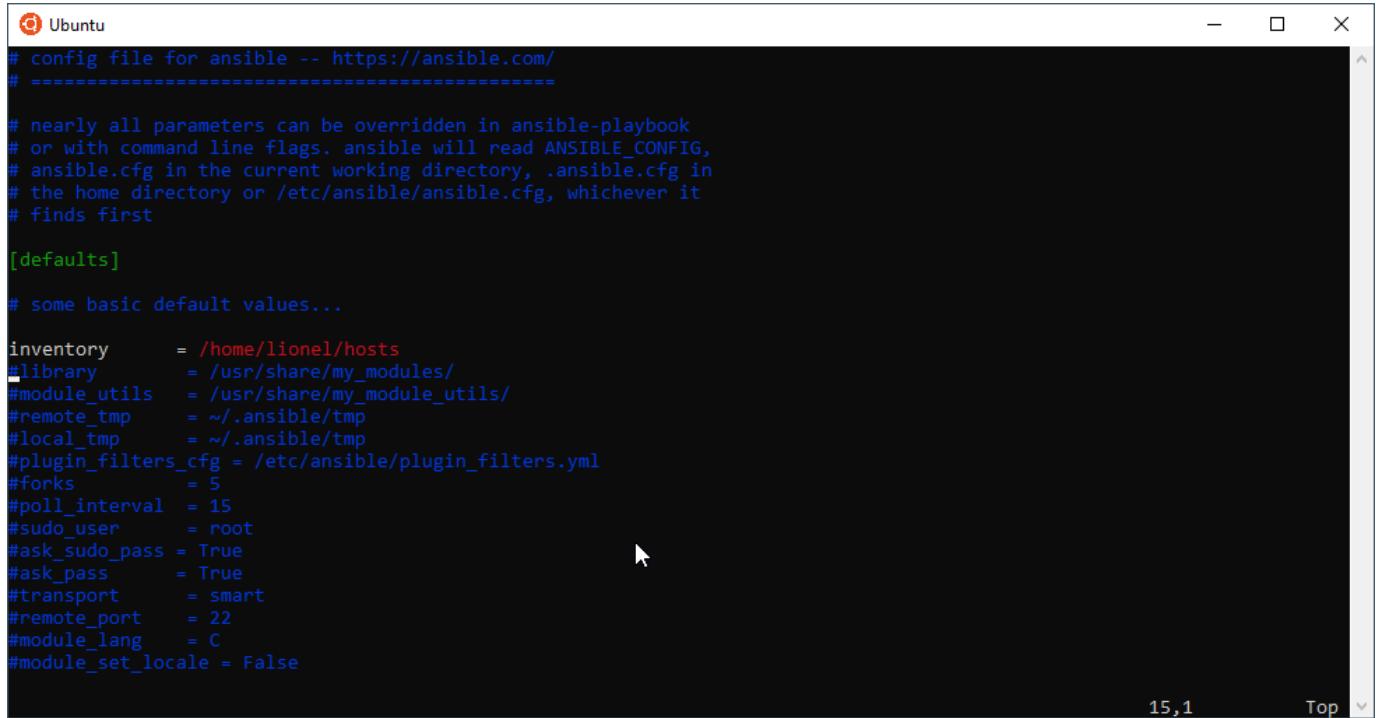
- Enter your password.
- Uncomment the inventory value:

A screenshot of a terminal window titled "Ubuntu". The window displays the contents of the file "/etc/ansible/ansible.cfg". A red arrow points to the line "#inventory = /etc/ansible/hosts", which is the line being uncommented. The rest of the configuration file shows various parameters and their values.

- 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



```
# 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]

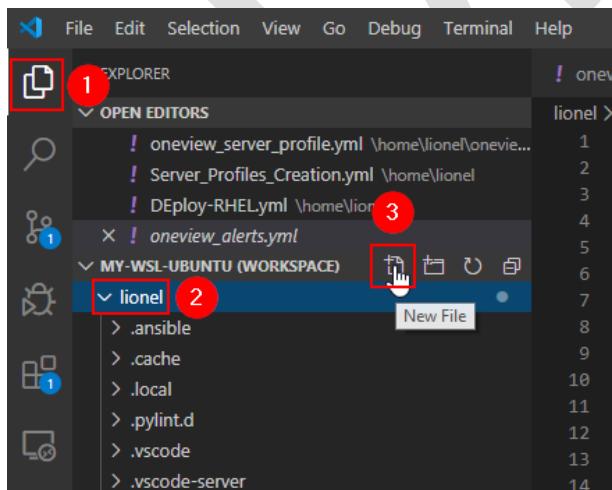
# some basic default values...

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
```

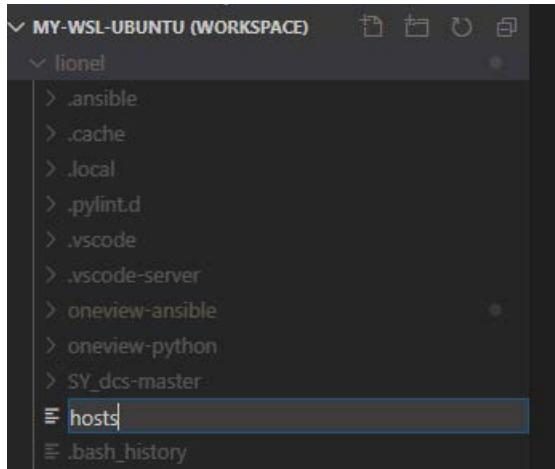
- Press **ESC** to exit *Insert Mode*, then type **:**(colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi

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

- Open VS Code, in the **Explorer** view, select your home folder then click on **New File**



- Named it **hosts**:

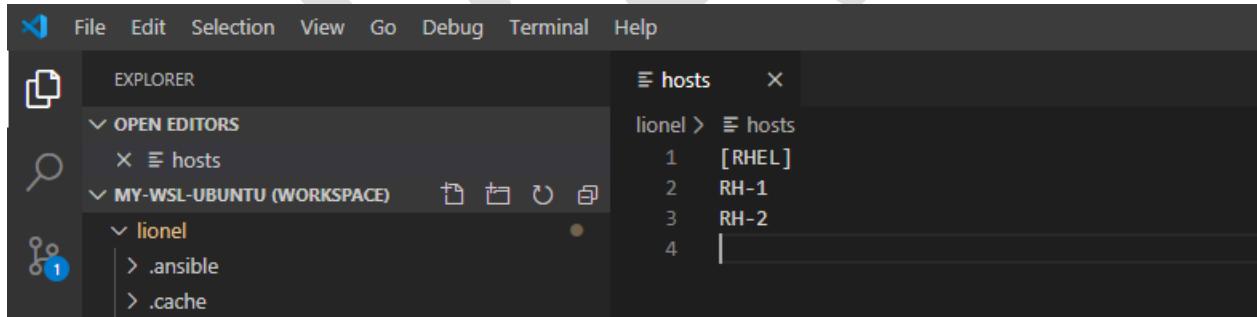


- Then add in the hosts file the following content:

```
[RHEL]
RH-1
RH-2
```

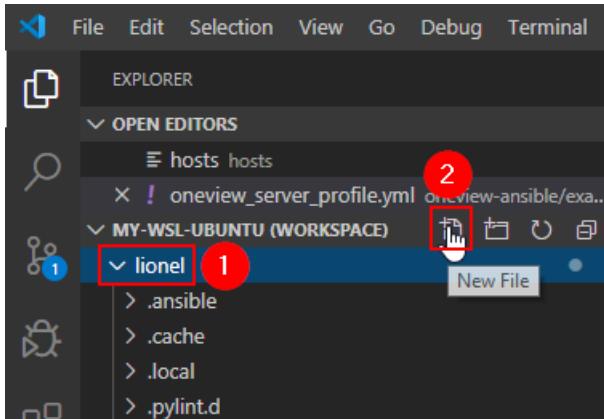
- Then save the file by pressing **CTRL + S**

Note: [...] defines *RHEL* as our group. *RH-1* and *RH-2* are the two systems defined in this group.

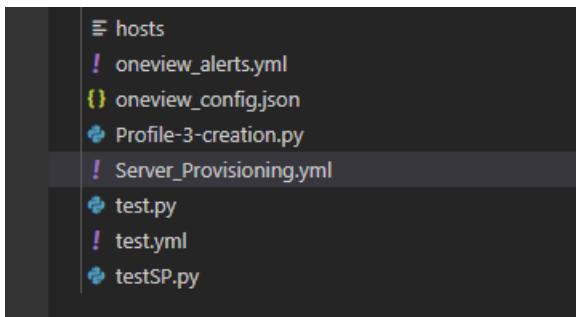


Now let's build a new playbook that will use this *hosts* file to create two server profiles (RH-1 and RH-2) from an existing Server Profile Template.

- Go back to VS Code, select your user home folder, click on **New File**



- Enter the name **Server_Provisioning.yml**



- Copy/paste the following playbook content:

```
---
- name: Making sure the Server Hardware tied to the Server Profile Template are turned off
  hosts: localhost
  gather_facts: no
  vars:
    - config: "{{ playbook_dir }}/oneview_config.json"
    - server_template: "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"

  tasks:
    - name: Gather facts about a Server Profile Template by name
      oneview_server_profile_template_facts:
        config: "{{ config }}"
        name: "{{ server_template }}"
        delegate_to: localhost

      # - debug: var=server_profile_templates

    - name: Get the Server Hardware Type Uri used by the template
      set_fact:
        serverHardwareTypeUri: "{{ server_profile_templates.0.serverHardwareTypeUri }}"

      # - debug: var=serverHardwareTypeUri

    - name: Get the Server Hardware using the SHT of the server profile template
      oneview_server_hardware_facts:
        config: "{{ config }}"
        delegate_to: localhost
      # - debug:
```

```

#     msg: "{{ server_hardwares | selectattr('serverHardwareTypeUri', 'equalto',
serverHardwareTypeUri ) | list | map(attribute='name') | list }}"
- set_fact:
    server_hardware_names : "{{ server_hardwares | selectattr('serverHardwareTypeUri', 'equalto',
serverHardwareTypeUri ) | list | map(attribute='name') | list }}"

# - debug: var=server_hardware_names

- name: Making sure the Compute Module(s) using the SHT are turned off
with_items: "{{ server_hardware_names }}"
oneview_server_hardware:
    config: "{{ config }}"
    state: power_state_set
    data:
        name : "{{ item }}"
        powerStateData:
            powerState: "Off"
            powerControl: "MomentaryPress"
    delegate_to: localhost

- name: Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template
in OneView
hosts: RHEL
gather_facts: no
vars:
- config: "{{ playbook_dir }}/oneview_config.json"
- server_template: "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"

tasks:
- name: Creating Server Profile [{{ inventory_hostname }}] from Server Profile Template [{{ server_template }}]
oneview_server_profile:
    config: "{{ config }}"
    data:
        serverProfileTemplateName: "{{ server_template }}"
        name: "{{ inventory_hostname }}"
    delegate_to: localhost
register: result

#- debug: var=server_hardware

- name: Task result of the Server Profile(s) creation
debug:
    msg: "{{ result.msg }}"

- name: Powering on the Compute Module(s) [{{ server.hardware.name }}]
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 }}"

```

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



This playbook contains two main group of tasks, one to turn off the server hardware and one to create the server profiles.

```
! 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...
48
49 > - name: Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template in OneView ...
85
```

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', serverHa
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
50

```

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:
        config: "{{ config }}"
        data:
          serverProfileTemplateName: "{{ server_template }}"
          name: "{{ inventory_hostname }}"
      delegate_to: localhost
      register: result

      #- debug: var=server_hardware

    - name: Task result of the Server Profile(s) creation
      debug:
        msg: "{{ result.msg }}"

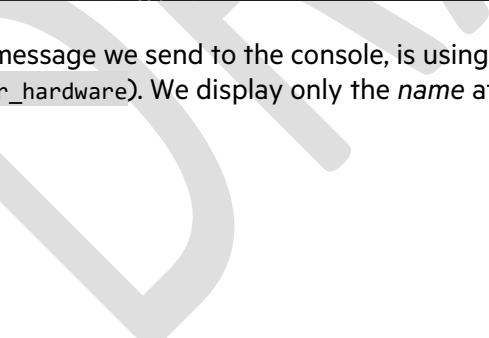
```

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`.

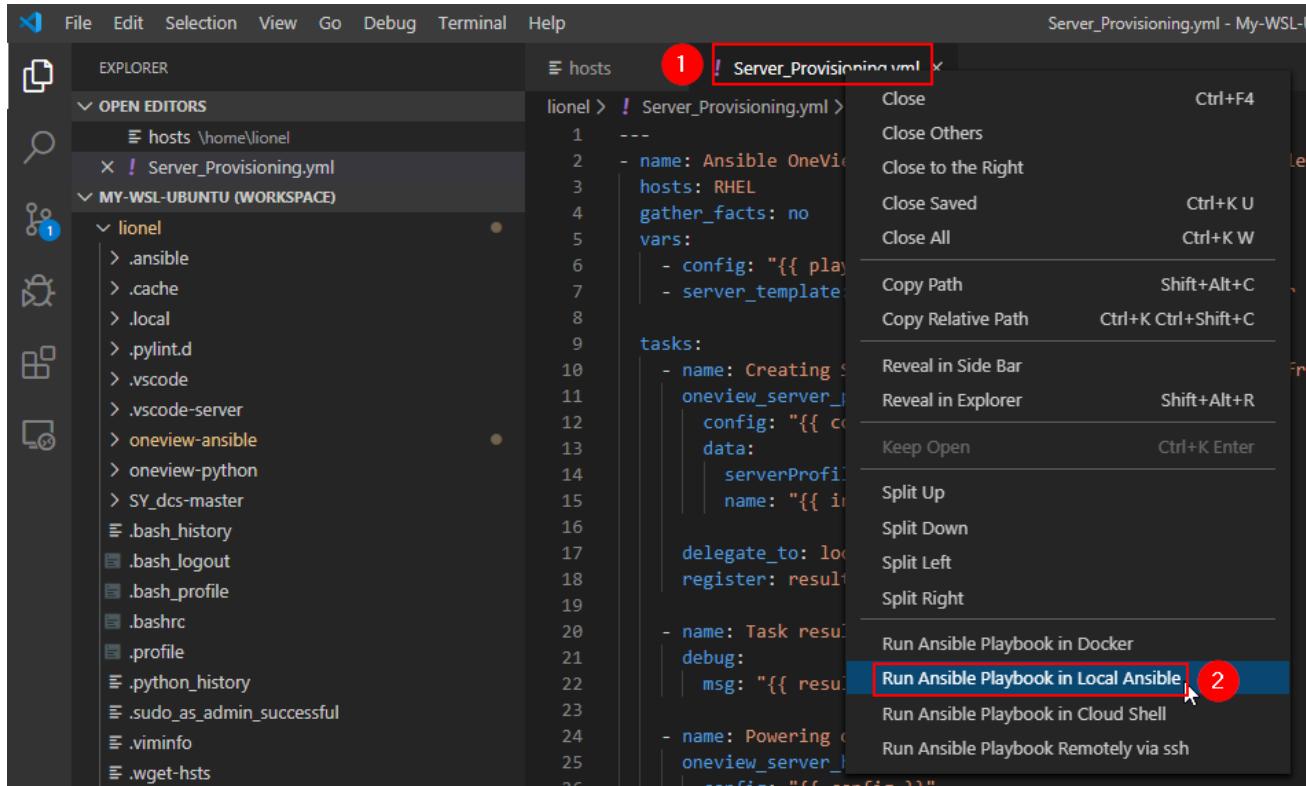


A screenshot of Visual Studio Code showing the file `oneview_server_profile.py`. The code defines several variables and their descriptions:

```
password: my_password
api_version: 1200
state: absent
data:
    name: Web-Server-L2
    delegate_to: localhost
    ...
RETURN = ...
server_profile:
    description: Has the OneView facts about the Server Profile.
    returned: On states 'present' and 'compliant'.
    type: dict
serial_number:
    description: Has the Server Profile serial number.
    returned: On states 'present' and 'compliant'.
    type: dict
server_hardware:
    description: Has the OneView facts about the Server Hardware.
    returned: On states 'present' and 'compliant'.
    type: dict
compliance_preview:
    description:
        Has the OneView facts about the manual and automatic updates required to make the server profile
        consistent with its template.
    returned: On states 'present' and 'compliant'.
    type: dict
created:
    description: Indicates if the Server Profile was created.
    returned: On states 'present' and 'compliant'.
    type: bool
...
import time
from copy import deepcopy
```

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	
RH-1	Description: Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows Server profile template: HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template Server hardware: Synergy-Encl-2.bay.3 Server hardware type: SY 660 Gen9 1 Enclosure group: EG-Synergy-Local Affinity: Off Device bay: VCGONC3006 Server power: fb43988d-7df3-4e6e-8619-c56eae9a32be Serial number (v): iqn.2015-02.com.hpe:oneview-vcg0nc3006 UUID (v): iSCSI initiator name (v):
RH-2	

- 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."
} ①

TASK [Powering on the Compute Module(s) [Synergy-Enc1-2, bay 3]] *****
changed: [RH-1] ②
changed: [RH-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. On the left, there's a sidebar titled "Server Profiles 5" with a green button "+ 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 "RH-1 General". The "General" tab is selected. 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 server hardware is a "SY 660 Gen9 1" in an "EG-Synergy-Local" enclosure group, located in "Synergy-Encl-2, bay 3". 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 after *Ansible – Scenario 2* when server profiles *RH-1* and *RH-2* are available in OneView.

- To do this, create a new file, named it **Server_Unprovisioning.yml** and add the following content:

```
---
- 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_hardware

    - 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

    - name: Deleting Server Profile [{{ inventory_hostname }}]
      oneview_server_profile:
        config: "{{ config }}"
        state: "absent"
        data:
          name: "{{ inventory_hostname }}"
      delegate_to: localhost
```

This playbook has three tasks:

- Getting server profiles information using the *hosts* file inventory information.
- Powering off the server hardware using the name collected from the server profile
- 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**



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

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Explorer:** Shows a tree view of files and folders. Under "MY-WSL-UBUNTU (WORKSPACE)", there is a folder "lionel" containing ".ansible", ".cache", ".local", ".pylint.d", ".vscode", ".vscode-server", "oneview-ansible", "oneview-python", "SY_dcs-master", ".bash_history", ".bash_logout", ".bash_profile", ".bashrc", ".profile", ".python_history", ".sudo_as_admin_successful", ".viminfo", ".wget-hsts", "Deploy-RHEL.yml", "hosts", "oneview_alerts.yml", "oneview_config.json", "Profile-3-creation.py", "Server_Provisioning.yml", "Server_Unprovisioning.yml", "test.py", "test.yml", and "testSP.py".
- Editor:** The main editor area displays a YAML file named "Server_Unprovisioning.yml". The file contains Ansible playbooks for provisioning and unprovisioning server profiles. It includes sections for hosts, vars, tasks, and oneview_server_profile.
- Contextual Menu:** A context menu is open over the file tab of "Server_Unprovisioning.yml". The menu items include Close, Close Others, Close to the Right, Close Saved, Close All, Copy Path, Copy Relative Path, Reveal in Side Bar, Reveal in Explorer, Keep Open, Split Up, Split Down, Split Left, Split Right, Run Ansible Playbook in Docker, Run Ansible Playbook in Local Ansible (which is highlighted with a red circle and a cursor), Run Ansible Playbook in Cloud Shell, and Run Ansible Playbook Remotely via ssh.

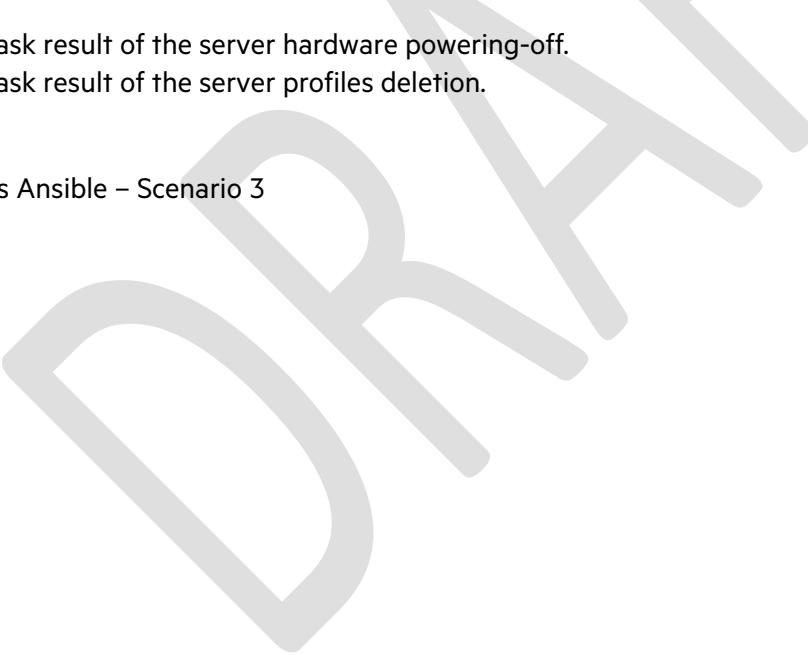
- Both servers should power-off and the server profiles should be erased.

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 green background. A red box highlights RH-2. At the top right, there is a 'General' tab with a dropdown menu. Below it, a progress bar indicates 'Clearing server hardware settings from Synergy-Encl-2, bay 3.' The main panel displays the 'General' configuration for RH-1, including fields for Description, Server profile template, Server hardware, Server hardware type, Enclosure group, Affinity, Server power, Serial number (v), UUID (v), and iSCSI initiator name (v).

- After a few seconds, the server profiles are deleted:

The screenshot shows the HPE OneView interface after the profiles have been deleted. The left sidebar now shows 'Server Profiles 0'. The main panel displays a message: 'Delete Completed 28s' with a timestamp of '2/25/20 11:13:57 am' and performed by 'administrator'. A large grey checkmark icon is overlaid on the bottom left of the screen.

- In the terminal, the following outputs get displayed:



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

lionel@MIC2FZV4RN:~$ export VS CODEEXT_USER_AGENT=vsco ss .vscode-ansible-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-1]
ok: [RH-1]

TASK [Powering off the server hardware [Synergy-Encl-2, bay 3]] ****
changed: [RH-1] ①
changed: [RH-2]

TASK [Deleting Server Profile [RH-1]] ****
changed: [RH-1] ②
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 – Execution plan to accelerate a configuration change

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 add a new network and assign this network to any of the existing Server Profiles resources using the network set *Prod*.

Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:

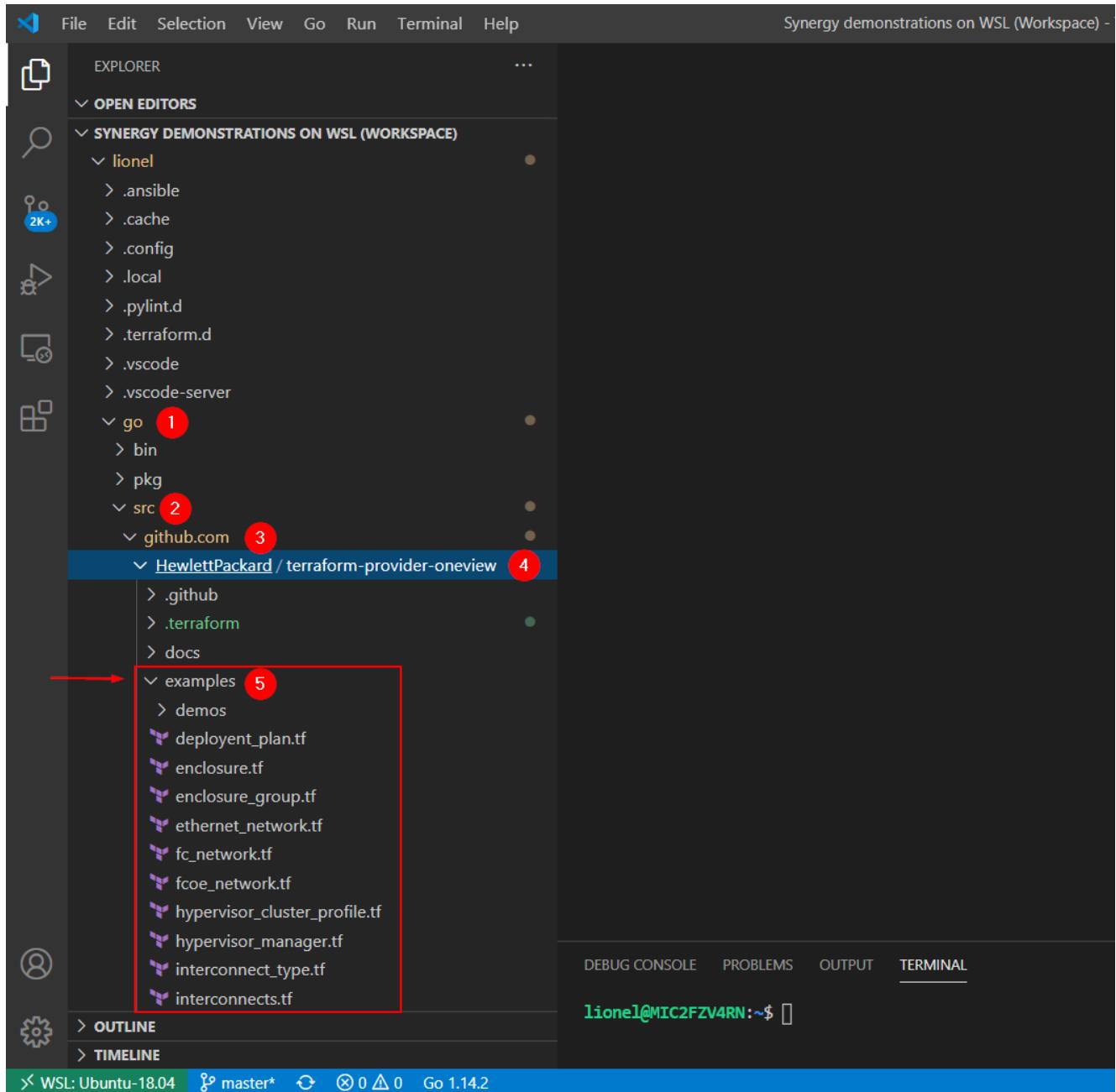


- **Important notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)

Like with the other HPE OneView partner ecosystem integrations, the HPE OneView Terraform provider offers many examples to facilitate the creation of Terraform configuration files.

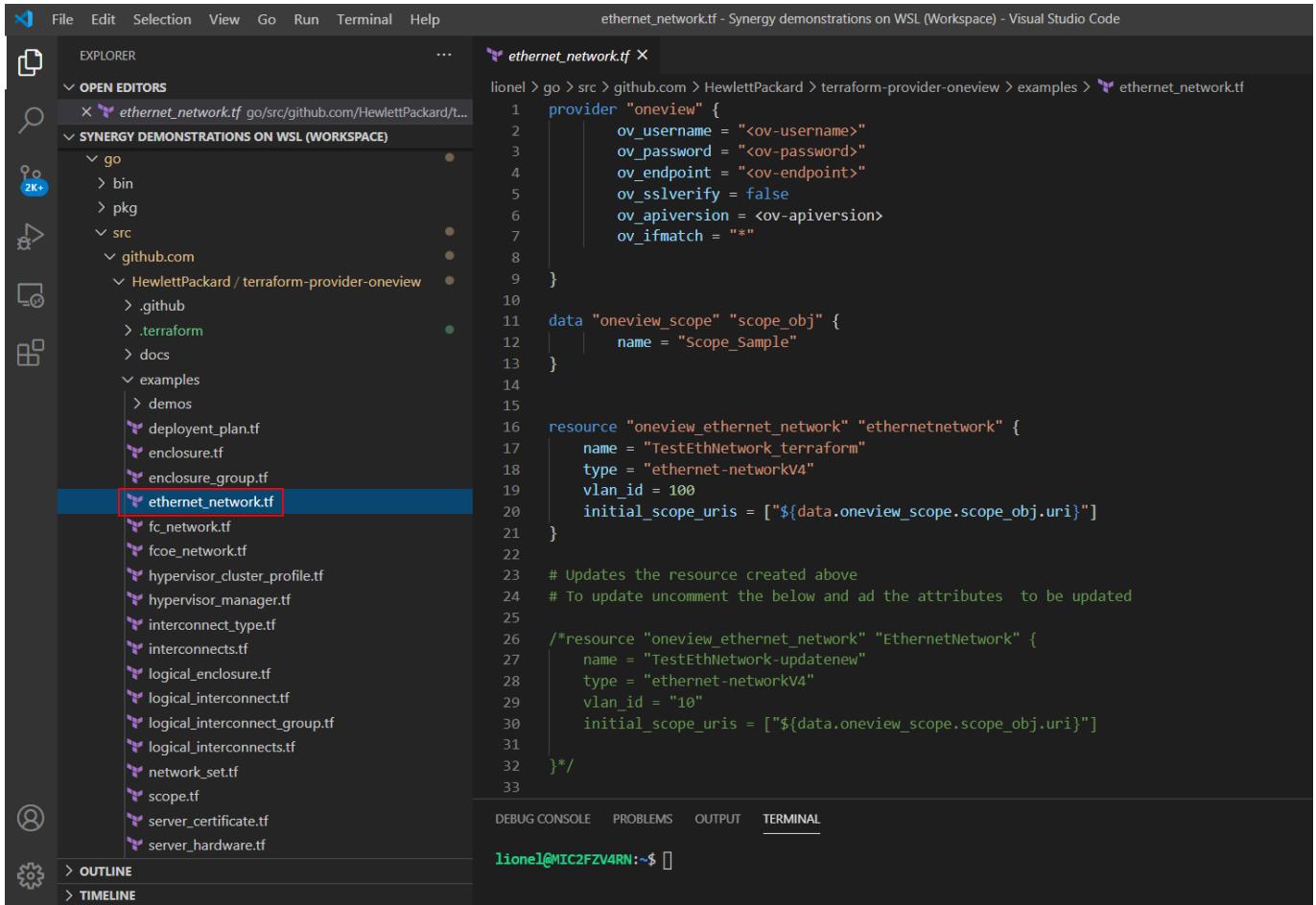
To access the Terraform examples:

- In Explorer, click on **go / src / github.com / HewlettPackard / terraform-provider-oneview** folder and browse the **examples** folder:



Many examples are provided by our *HPE OneView-terraform* library. Each OneView main resource operation is exposed through a Terraform configuration file using the HashiCorp Configuration Language (HCL).

- Select **ethernet_network.tf**



```
provider "oneview" {
    ov_username = "<ov-username>"
    ov_password = "<ov-password>"
    ov_endpoint = "<ov-endpoint>"
    ov_sslverify = false
    ov_apiversion = <ov-apiversion>
    ov_ifmatch = "*"
}

data "oneview_scope" "scope_obj" {
    name = "Scope_Sample"
}

resource "oneview_ethernet_network" "ethernetnetwork" {
    name = "testEthNetwork_terraform"
    type = "ethernet-networkV4"
    vlan_id = 100
    initial_scope_uris = ["${data.oneview_scope.scope_obj.uri}"]
}

# Updates the resource created above
# To update uncomment the below and add the attributes to be updated
/*resource "oneview_ethernet_network" "EthernetNetwork" {
    name = "TestEthNetwork-updatenew"
    type = "ethernet-networkV4"
    vlan_id = "10"
    initial_scope_uris = ["${data.oneview_scope.scope_obj.uri}"]
}*/
```

This Terraform configuration file is an example for creating and updating an ethernet network.

- Useful documentation is also available in **/docs**:

1. **/docs/d** = folder documentation on how to access the attribute of a resource (**d** for data source)
2. **/docs/r** = folder documentation on how to create a resource (**r** for resource)

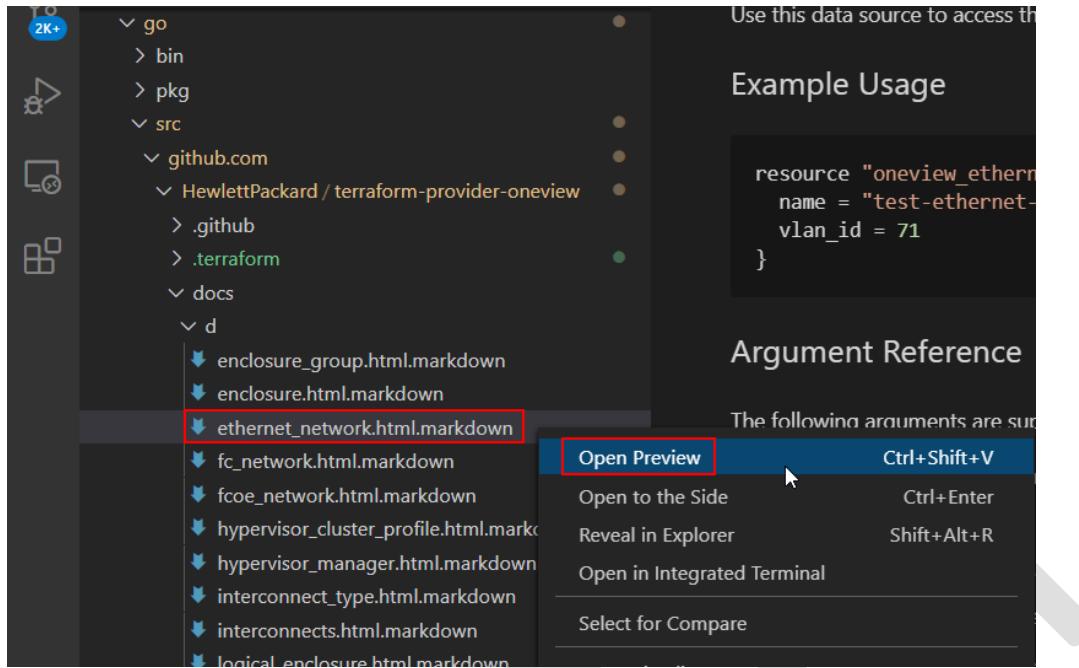
The screenshot shows a dark-themed code editor interface. On the left is the Explorer sidebar, which lists several folders and files under the 'SYNTERGY DEMONSTRATIONS ON WSL (WORKSPACE)' section. A red box highlights the 'docs' folder, which contains sub-folders 'd' and 'r'. To the right is the main editor area displaying the content of the 'ethernet_network.tf' file. The code defines a provider for OneView and a data source named 'oneview'. It also defines a resource 'oneview_enclosure' with attributes like 'name', 'type', 'voltage', and 'initial_size'.

```
provider "oneview" {
    ov_u
    ov_p
    ov_e
    ov_s
    ov_a
    ov_i
}

data "oneview_enclosure" {
    name
}

resource "oneview_enclosure" {
    name = ""
    type = ""
    voltage = ""
    vlan_id = ""
    initial_size = ""
}
```

- Select **ethernet_network.html.markdown** in **/docs/d/**



- You can right click and select **Open Preview** or press **Ctrl+Shift+V** to get a preview of the markdown file:

The screenshot shows a browser window with a dark theme. The title bar says 'Preview ethernet_network.html.markdown'. The page content includes:
- A header: 'oneview_ethernet_network'
- A note: 'Use this data source to access the attributes of an Ethernet network.'
- A section: 'Example Usage'
- A code snippet:

```
resource "oneview_ethernet_network" "default" {
  name = "test-ethernet-network"
  vlan_id = 71
}
```


- A section: 'Argument Reference'
- A note: 'The following arguments are supported:'
- A list of supported arguments:

- **name** - (Required) A unique name for the resource.
- **vlan_id** - (Required) The Virtual LAN (VLAN) identification number (integer) assigned to the network. Changing this forces a new resource.
- **purpose** - (Optional) A description of the network's role within the logical interconnect. This defaults to General.
- **private_network** - (Optional) When enabled, the network is configured so that all downlink (server) ports connected to the network are prevented from communicating with each other within the logical interconnect. This defaults to false.
- **smart_link** - (Optional) When enabled, the network is configured so that, within a logical interconnect, all uplinks that carry the network are monitored. This defaults to false.
- **ethernet_network_type** - (Optional) The type of Ethernet network. This defaults to Tagged.

This page shows the various supported arguments that we can use with the `oneview_ethernet_network` resource that were not indicated in the configuration file example.

So, let's start by creating a new Terraform project which will host our configuration file that will provide all the required actions to add a new network in OneView and assign this network to any of the existing Server Profiles resources.

- Open the WSL Ubuntu console and go back to your home directory, 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.

The *terraform init* command will display the following message if run from the VC Code console:

```
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/username/terraform-create-profile/create-profile.tf: At 2:9: illegal char

- Then create a new directory named *terraform-create-profile*

```
mkdir terraform-create-profile
```

- Move into the directory:

```
cd terraform-create-profile
```

- Then type:

```
touch create-network.tf
```

```
lionel@MIC2FZV4RN: ~/terraform-create-profile
lionel@MIC2FZV4RN:~/terraform-create-profile$ cd ~
lionel@MIC2FZV4RN:~$ mkdir terraform-create-network
lionel@MIC2FZV4RN:~$ cd terraform-create-network/
lionel@MIC2FZV4RN:~/terraform-create-network$ touch create-network.tf
lionel@MIC2FZV4RN:~/terraform-create-network$
```

- Then let's create the Terraform configuration file using:

```
cat > create-network.tf << 'EOF'

provider "oneview" {
  ov_domain = "Local"
  ov_username = "Administrator"
  ov_password = "password"
  ov_endpoint = "https://192.168.56.101"
  ov_sslverify = false
```

```

ov_apiversion = 1600
ov_ifmatch = "*"
}

// 1 - IMPORTING RESOURCE FIRST

resource "oneview_network_set" "network_set" {
}

// resource "oneview_logical_interconnect_group" "logical_interconnect_group" {
// }

// 2 - ADDING THE NEW NETWORK

// CREATION OF ETHERNET NETWORK
# resource "oneview_ethernet_network" "network_Prod_60" {
#   name = "Prod_60"
#   type = "ethernet-networkV4"
#   vlan_id = 60
# }

# /* GETTING ETHERNET NETWORK URIS FROM THE NETWORK SET */
# data "oneview_network_set" "network_set_Prod" {
#   name = "Prod"
# }

# // output "network_set_Prod_uris" {
# //   value = "${data.oneview_network_set.network_set_Prod.network_uris}"
# // }

# /* GETTING NETWORK URIs FOR THE LIG DECLARATION*/
# data "oneview_ethernet_network" "network_Mgmt" {
#   name = "Mgmt"
# }
# data "oneview_fc_network" "network_SAN_A_FC" {
#   name = "SAN A FC"
# }
# data "oneview_fc_network" "network_SAN_B_FC" {
#   name = "SAN B FC"
# }

# // output "network_SAN_B_FC_uri" {
# //   value = "${data.oneview_fc_network.network_SAN_B_FC.uri}"
# // }

# /* GETTING THE UPLINK SET URIs FOR THE LIG DECLARATION*/
# data "oneview_uplink_set" "uplink_set_Mgmt" {
#   name = "Mgmt"
# }
# // output "oneview_uplink_set_value" {
# //   value = "${data.oneview_uplink_set.uplink_set_Mgmt.port_config_infos}"
# // }

# // ADDING NEW NETWORK URI TO LOGICAL INTERCONNECT GROUP UPLINK SET URIS
# resource "oneview_logical_interconnect_group" "logical_interconnect_group" {

```

```
# type = "logical-interconnect-groupV8"
#     interconnect_bay_set = 3
#     enclosure_indexes = [1, 2, 3]
# redundancy_type = "HighlyAvailable"
# name = "LIG-FlexFabric"
# internal_network_uris = []
# igmp_settings = []
# sflow_configuration = []
#     interconnect_map_entry_template = [{}
#     enclosure_index = 1
#     bay_number = 3
#     interconnect_type_name = "Virtual Connect SE 40Gb F8 Module for Synergy"
# ],
# {
#     enclosure_index = 2
#     bay_number = 6
#     interconnect_type_name = "Virtual Connect SE 40Gb F8 Module for Synergy"
# },
# {
#     enclosure_index = 3
#     bay_number = 3
#     interconnect_type_name = "Synergy 20Gb Interconnect Link Module"
# },
# {
#     enclosure_index = 3
#     bay_number = 6
#     interconnect_type_name = "Synergy 20Gb Interconnect Link Module"
# },
# {
#     enclosure_index = 1
#     bay_number = 6
#     interconnect_type_name = "Synergy 20Gb Interconnect Link Module"
# },
# {
#     enclosure_index = 2
#     bay_number = 3
#     interconnect_type_name = "Synergy 20Gb Interconnect Link Module"
# }]
# uplink_set = [
# {
#     name = "Mgmt"
#     network_type = "Ethernet"
#     lacp_timer = "Short"
#     mode = "Auto"
#     network_uris = ["${data.oneview_ethernet_network.network_Mgmt.uri}"]
#     logical_port_config = [
#         {
#             port_num = [63]
#             bay_num = 6
#             enclosure_num = 2
#             primary_port = "false"
#             desired_speed = "Auto"
#         },
#         {
#             port_num = [63]
#             bay_num = 3
#             enclosure_num = 1
#             primary_port = "false"
#             desired_speed = "Auto"
#         }
#     ]
# }
```

```

# },
# {
#     name = "Prod"
#     network_type = "Ethernet"
#     lacp_timer = "Short"
#     mode = "Auto"
#         network_uris =
["${data.oneview_network_set.network_set_Prod.network_uris}","${oneview_ethernet_network.network_Prod_60
.uri}"]
#     logical_port_config = [
#         {
#             port_num = [65]
#             bay_num = 3
#             enclosure_num = 1
#             primary_port = "false"
#             desired_speed = "Auto"
#         },
#         {
#             port_num = [65]
#             bay_num = 6
#             enclosure_num = 2
#             primary_port = "false"
#             desired_speed = "Auto"
#         }
#     ]
# },
# {
#     name = "SAN-B-FC"
#     network_type = "FibreChannel"
#     network_uris = ["${data.oneview_fc_network.network_SAN_B_FC.uri}"]
#     logical_port_config = [
#         {
#             port_num = [67]
#             bay_num = 6
#             enclosure_num = 2
#             primary_port = "false"
#             desired_speed = "Auto"
#         }
#     ]
# },
# {
#     name = "SAN-A-FC"
#     network_type = "FibreChannel"
#     network_uris = ["${data.oneview_fc_network.network_SAN_A_FC.uri}"]
#     logical_port_config = [
#         {
#             port_num = [67]
#             bay_num = 3
#             enclosure_num = 1
#             primary_port = "false"
#             desired_speed = "Auto"
#         }
#     ]
# }
# ]
# depends_on = ["oneview_ethernet_network.network_Prod_60"]
# }

# /* GETTING THE LI URI*/
# // data "oneview_logical_interconnect" "logical_interconnect_read" {

```

```

# //      name = "LE-Synergy-Local-LIG-FlexFabric"
# // }

# // PERFORMING UPDATE FROM GROUP ON LOGICAL INTERCONNECT TO BRING BACK IT TO CONSISTENT STATE
# resource "oneview_logical_interconnect" "logical_interconnect" {
#   update_type = "updateComplianceById"
#   //uri = "${data.oneview_logical_interconnect.logical_interconnect_read.uri}"
#   depends_on = ["oneview_logical_interconnect_group.logical_interconnect_group"]
# }

# // ADDING THE NEW NETWORK TO THE NETWORK SET
# resource "oneview_network_set" "network_set" {
#   name = "Prod"
#   native_network_uri = ""
#   network_set_type = "Regular"
#   type = "network-setV5"
#   network_uris =
["${data.oneview_network_set.network_set_Prod.network_uris}","${oneview_ethernet_network.network_Prod_60
.uri}"]
#   depends_on = ["oneview_logical_interconnect.logical_interconnect"]
# }

EOF

```

- Once the creation of the configuration file is complete, you can move back to the VS Code console and open this new Terraform file to get a better display with syntax highlighting:

```

provider "oneview" {
  ov_domain = "Local"
  ov_username = "Administrator"
  ov_password = "password"
  ov_endpoint = "https://192.168.56.101"
  ov_sslverify = false
  ov_apiversion = 1600
  ov_ifmatch = "*"
}

// CREATION OF ETHERNET NETWORK
resource "oneview_ethernet_network" "ethernet_network" {
  name = "Prod_60"
  type = "ethernet-networkV4"
  vlan_id = 60
}

// DISPLAYING NEW NETWORK URI
output "oneview_ethernet_network_uri" {
  value = "${oneview_ethernet_network.ethernet_network.uri}"
}

// GETTING THE NETWORK SET DATA TO POPULATE LATER THE LIG UPLINK SET URIS IN USE
data "oneview_network_set" "network_set" {
  name = "Prod"
}

// DISPLAYING EXISTING NETWORK SET URIS
output "existing_network_set_value" {
  value = "${data.oneview_network_set.network_set.uri}"
}

```

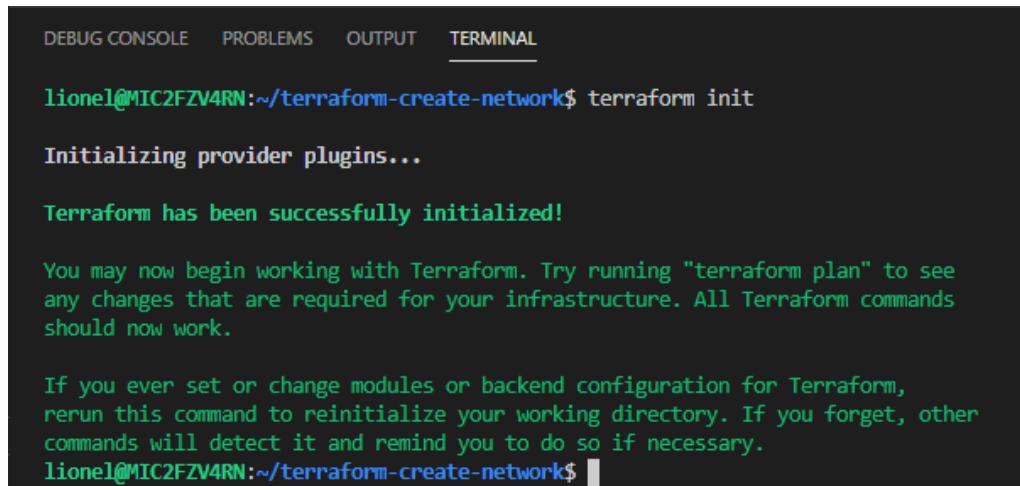
- From the VS Code console, move into the `terraform-create-network` folder:

```
cd ~/terraform-create-network
```

- The first step is to run the *init* command from the project's directory:

```
terraform init
```

This command downloads the *oneview* provider plugin and take other actions needed to initialize our project.



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

lionel@MIC2FZV4RN:~/terraform-create-network$ terraform init
Initializing provider plugins...
Terraform has been successfully initialized!

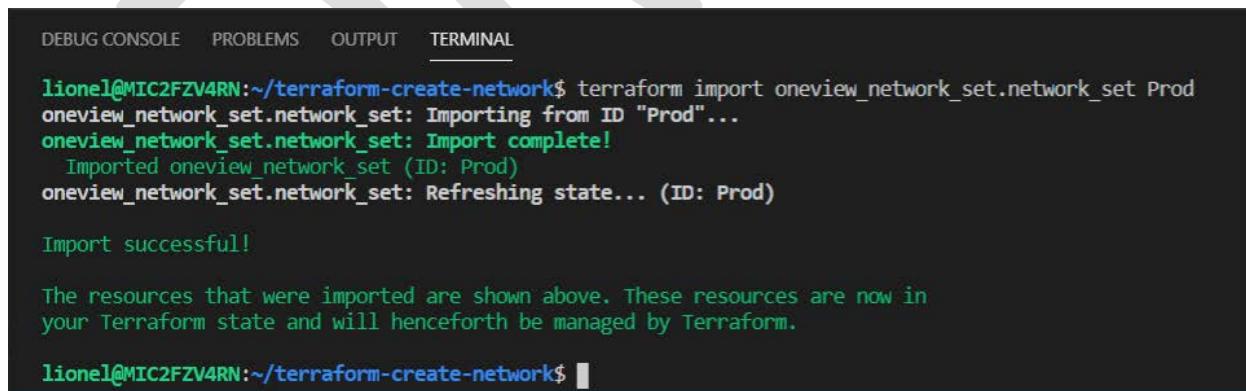
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

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-network$
```

Note: This command can be run more than once, but you generally will only need to run it again if you are adding another provider to your project.

- Next, we need to import existing Network Set, LIG and LI resources from appliance into the Terraform instances so that we make updates to these resources later:

```
terraform import oneview_network_set.network_set Prod
```



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

lionel@MIC2FZV4RN:~/terraform-create-network$ terraform import oneview_network_set.network_set Prod
oneview_network_set.network_set: Importing from ID "Prod"...
oneview_network_set.network_set: Import complete!
  Imported oneview_network_set (ID: Prod)
oneview_network_set.network_set: Refreshing state... (ID: Prod)

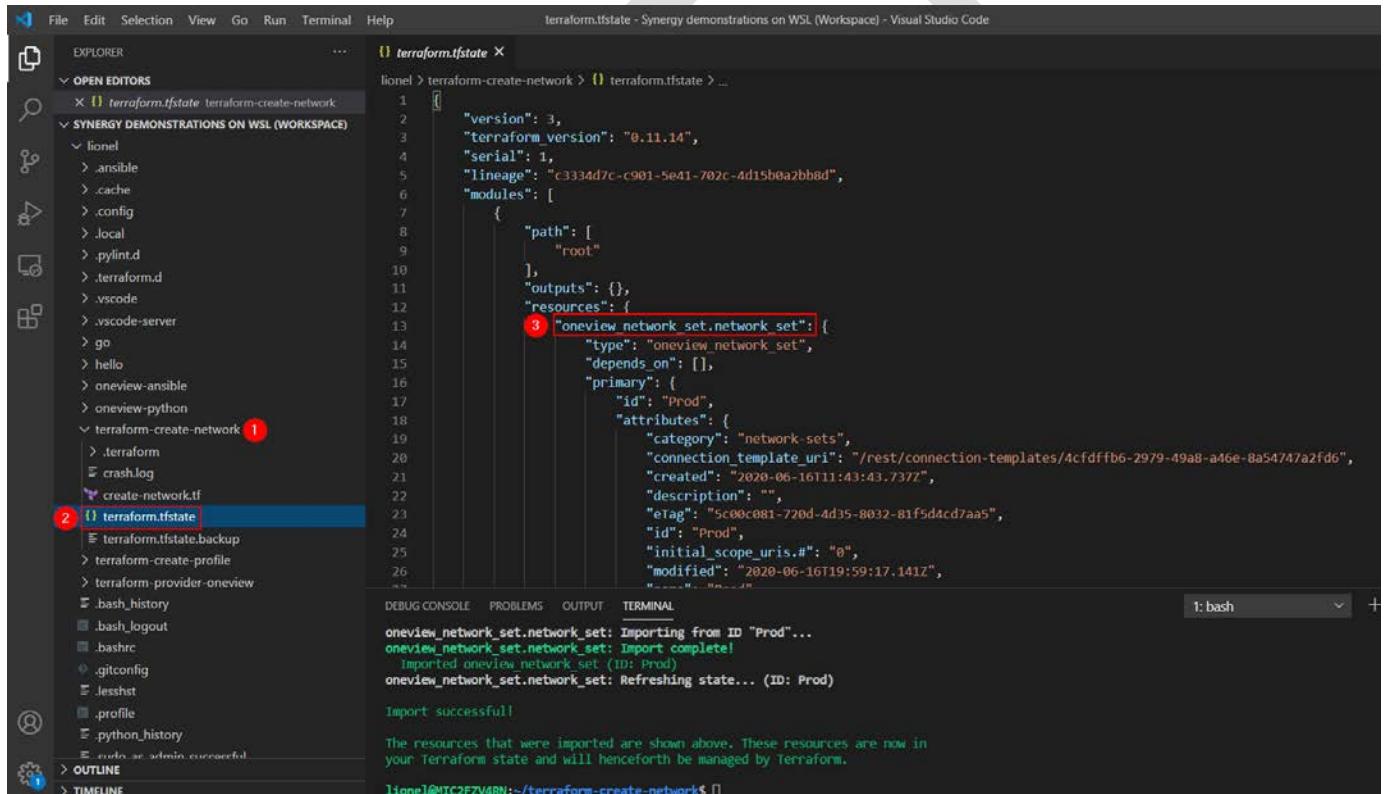
Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.

lionel@MIC2FZV4RN:~/terraform-create-network$
```

Note: this import command uses the first resource block in our terraform file where we have defined the network set resource.

- Notice that a new file `terraform.tfstate` has been generated in the project folder which now contains the network set resource with its various attributes:



- We also need to import the LIG and LI resources. To import the LIG, we can use our FlexFabric LIG name:

```
terraform import oneview_logical_interconnect_group.logical_interconnect_group LIG-FlexFabric
```

```
lionel@MIC2FZV4RN:~/terraform-create-network$ terraform import oneview_logical_interconnect_group.logical_interconnect_group LIG-FlexFabric
oneview_logical_interconnect_group.logical_interconnect_group: Importing from ID "LIG-FlexFabric"...
oneview_logical_interconnect_group.logical_interconnect_group: Import complete!
  Imported oneview_logical_interconnect_group (ID: LIG-FlexFabric)
oneview_logical_interconnect_group.logical_interconnect_group: Refreshing state... (ID: LIG-FlexFabric)

Import successful!
```

The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform.

```
lionel@MIC2FZV4RN:~/terraform-create-network$
```

Note: The Terraform import command currently can only import one resource at a time. This means we cannot yet point Terraform import to an entire collection of resources such as Logical Interconnects and import all of them.

- Notice this new LIG resource with its long list of attributes has been added to the *terraform.tfstate* file:

```
! terraform.tfstate ×
lionel > terraform-create-network > ! terraform.tfstate > ...
1  {
2    "version": 3,
3    "terraform_version": "0.11.14",
4    "serial": 2,
5    "lineage": "c3334d7c-c901-5e41-702c-4d15b0a2bb8d",
6    "modules": [
7      {
8        "path": [
9          "root"
10        ],
11        "outputs": {},
12        "resources": {
13          "oneview_logical_interconnect_group.logical_interconnect_group": {
14            "type": "oneview_logical_interconnect_group",
15            "depends_on": [],
16            "primary": {
17              "id": "LIG-FlexFabric",
18              "attributes": {
19                "category": "logical-interconnect-groups",
20                "created": "2020-06-16T11:45:48.729Z",
21                "description": "",
22                "eTag": "9e45cce5-5870-4ac1-8b69-d8db84628820",
23                "enclosure_indexes.#": "3",
24                "enclosure_indexes.1": "1",
25                "enclosure_indexes.2": "2",
26                "enclosure_indexes.3": "3",
27              }
28            }
29          }
30        }
31      }
32    }
33  }
```

- To import the Logical Interconnect resource, we need to provide the LI ID and not the LI name as for the LIG. This fact is found in the provided examples below:

```

File Edit Selection View Go Run Terminal Help
logical_interconnect.tf - Synergy demonstrations on WSL (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
create-network.tf synergy_infrastructure_provisioning_network.tf logical_interconnect.tf
lionel > terraform-provider-oneview > examples > logical_interconnect.tf
1 provider "oneview" {
2   ov_username = "<ov_username>"
3   ov_password = "<ov_password>"
4   ov_endpoint = "<ov_endpoint>"
5   ov_sslverify = false
6   ov_apiversion = <ov_apiversion>
7   ov_ifmatch = "*"
8 }
9
10 data "oneview_logical_interconnect" "logical_interconnect" {
11   name = "d4468f89-4442-4324-9c01-624c7382db2d" 3
12 }
13
14 output "oneview_logical_interconnect_value" {
15   value = "${data.oneview_logical_interconnect.logical_interconnect.uri}"
16 }
17

```

- So, to import the Logical Interconnect resource of our VC FlexFabric modules, we need first to find its ID. To do this, we can browse the OneView UI and go to the Logical Interconnects page and copy the ID found in the URL when we select LE-Synergy-Local-LIG-FlexFabric:

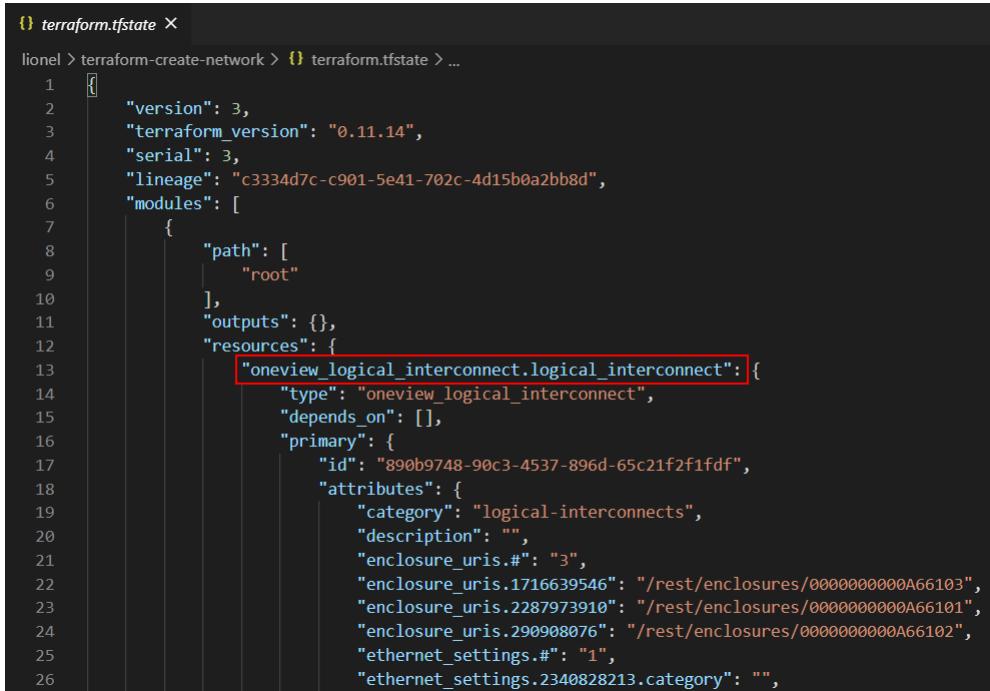
Name
LE-Synergy-Local-LIG-FC-1
LE-Synergy-Local-LIG-FC-2
LE-Synergy-Local-LIG-FC-3
LE-Synergy-Local-LIG-FlexFabric
LE-Synergy-Local-LIG-SAS-1
LE-Synergy-Local-LIG-SAS-2
LE-Synergy-Local-LIG-SAS-3

Note: The ID is starting just after **/rest/logical-interconnects/** and ending before the **?**.
For instance, the ID in this URL below is 890b9748-90c3-4537-896d-65c21f2f1fdf:
https://192.168.56.101/#/logicalswitch/show/switch/r/rest/logical-interconnects/890b9748-90c3-4537-896d-65c21f2f1fdf?_q=scope%3Aall_resources&f_

- Then we can use the import command using this ID:

```
terraform import oneview_logical_interconnect.logical_interconnect 890b9748-90c3-4537-896d-65c21f2f1fdf
```

- Once again *terraform.tfstate* gets updated with the LI resource:



```

 1  {
 2      "version": 3,
 3      "terraform_version": "0.11.14",
 4      "serial": 3,
 5      "lineage": "c3334d7c-c901-5e41-702c-4d15b0a2bb8d",
 6      "modules": [
 7          {
 8              "path": [
 9                  "root"
10              ],
11              "outputs": {},
12              "resources": {
13                  "oneview_logical_interconnect.logical_interconnect": [
14                      {
15                          "type": "oneview_logical_interconnect",
16                          "depends_on": [],
17                          "primary": {
18                              "id": "890b9748-90c3-4537-896d-65c21f2f1fdf",
19                              "attributes": {
20                                  "category": "logical-interconnects",
21                                  "description": "",
22                                  "enclosure_uris.#": "3",
23                                  "enclosure_uris.1716639546": "/rest/enclosures/000000000A66103",
24                                  "enclosure_uris.2287973910": "/rest/enclosures/000000000A66101",
25                                  "enclosure_uris.290908076": "/rest/enclosures/000000000A66102",
26                                  "ethernet_settings.#": "1",
27                                  "ethernet_settings.2340828213.category": ""
28                              }
29                          }
30                      ]
31                  }
32              }
33          }
34      ]
35  }

```

- Once the 3 resources import is complete, we need to comment the first section of the configuration file and uncomment the second one. The file will be changed as follows:

Before the change

```

// 1 - IMPORTING RESOURCE FIRST

resource "oneview_network_set" "network_set" {}

resource "oneview_logical_interconnect_group" "logical_interconnect_group" {}

resource "oneview_logical_interconnect" "logical_interconnect" {}

// 2 - ADDING THE NEW NETWORK

// CREATION OF ETHERNET NETWORK
# resource "oneview_ethernet_network" "ethernet_network" {
#   name = "Prod_60"
#   type = "ethernet-networkV4"
#   vlan_id = 60
# }

# // ADDING NEW NETWORK URI TO LOGICAL INTERCONNECT GROUP UPLINK SET URIS
# resource "oneview_logical_interconnect_group" "logical_interconnect_group" {
#   type = "logical-interconnect-groupV8"
#   interconnect_bay_set = 3
#   enclosure_indexes = [1, 2, 3]
#   redundancy_type = "HighlyAvailable"
#   name = "LIG-FlexFabric"
#   internal_network_uris = []
#   interconnect_map_entry_template = [
#     {
#       enclosure_index = 1
#       bay_number = 3
#       interconnect_type_name = "Virtual Connect SE 40Gb F8 Module for Synergy"
#     },

```

After the change

```

// 1 - IMPORTING RESOURCE FIRST

# resource "oneview_network_set" "network_set" {}
# 

# resource "oneview_logical_interconnect_group" "logical_interconnect_group" {}
# 

# resource "oneview_logical_interconnect" "logical_interconnect" {}
# 

// 2 - ADDING THE NEW NETWORK

// CREATION OF ETHERNET NETWORK
resource "oneview_ethernet_network" "ethernet_network" {
  name = "Prod_60"
  type = "ethernet-networkV4"
  vlan_id = 60
}

// ADDING NEW NETWORK URI TO LOGICAL INTERCONNECT GROUP UPLINK SET URIS
resource "oneview_logical_interconnect_group" "logical_interconnect_group" [
  type = "logical-interconnect-groupV8"
  interconnect_bay_set = 3
  enclosure_indexes = [1, 2, 3]
  redundancy_type = "HighlyAvailable"
  name = "LIG-FlexFabric"
  internal_network_uris = []
  interconnect_map_entry_template = [
    {
      enclosure_index = 1
      bay_number = 3
      interconnect_type_name = "Virtual Connect SE 40Gb F8 Module for Synergy"
    },

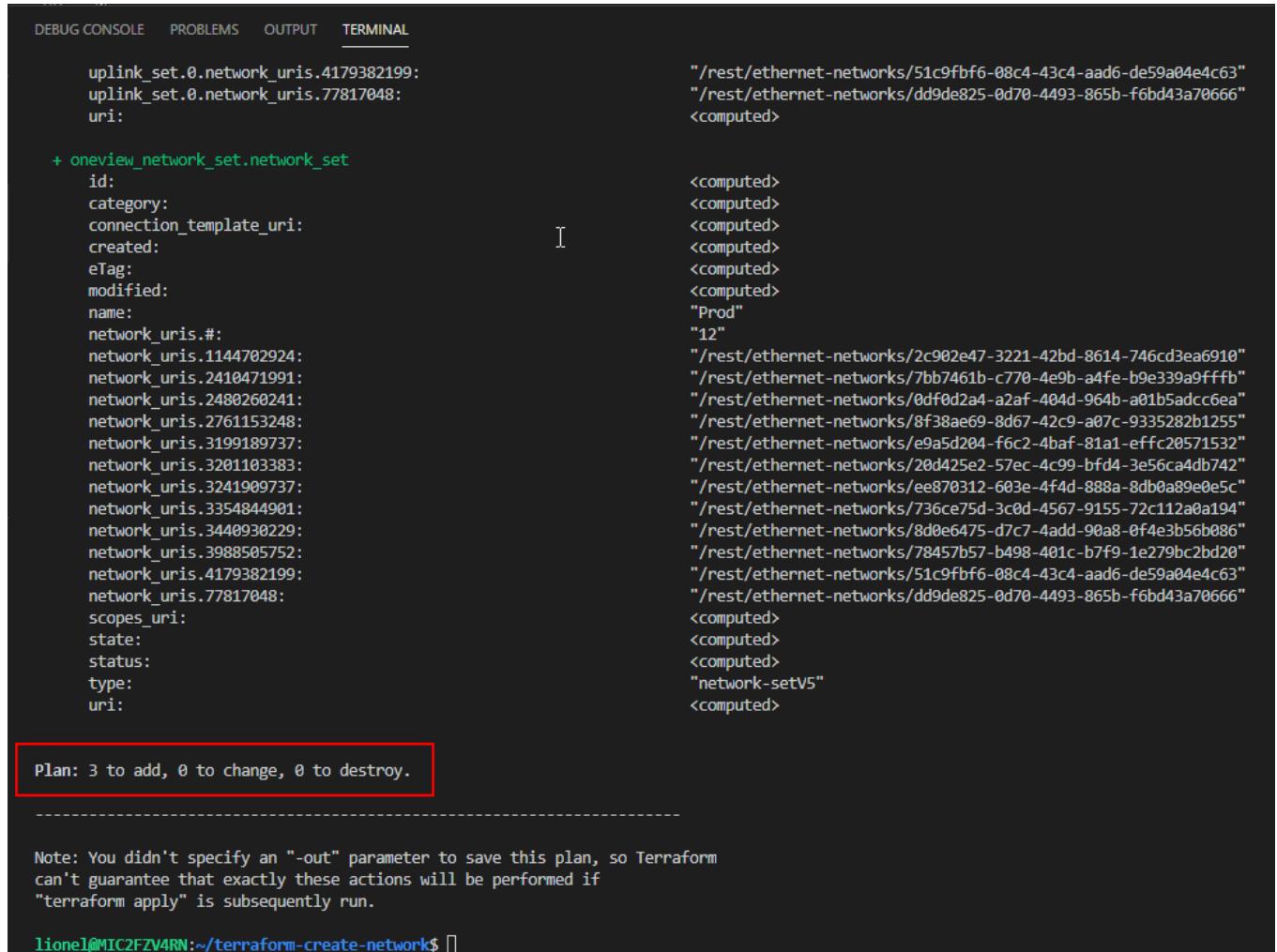
```

Note: In VS Code, you can comment/uncomment a selected section using **Ctrl + /**

- Save the file then run the first plan to see what changes Terraform has pending:

```
terraform plan
```

- This command generates a report detailing what actions Terraform will take to set up our OneView resources. We can see that Terraform will perform 1 action to add, 2 to change and 0 to destroy



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

uplink_set.0.network_uris.4179382199:
uplink_set.0.network_uris.77817048:
uri:

+ oneview_network_set.network_set
  id: <computed>
  category: <computed>
  connection_template_uri: <computed>
  created: <computed>
  eTag: <computed>
  modified: <computed>
  name: "Prod"
  network_uris.#: "12"
  network_uris.1144702924: "/rest/ethernet-networks/2c902e47-3221-42bd-8614-746cd3ea6910"
  network_uris.2410471991: "/rest/ethernet-networks/7bb7461b-c770-4e9b-a4fe-b9e339a9fffb"
  network_uris.2488260241: "/rest/ethernet-networks/0df0d2a4-a2af-404d-964b-a01b5adcc6ea"
  network_uris.2761153248: "/rest/ethernet-networks/8f38ae69-8d67-42c9-a07c-9335282b1255"
  network_uris.3199189737: "/rest/ethernet-networks/e9a5d204-f6c2-4baf-81a1-effc20571532"
  network_uris.3201103383: "/rest/ethernet-networks/20d425e2-57ec-4c99-bfd4-3e56ca4db742"
  network_uris.3241909737: "/rest/ethernet-networks/ee870312-603e-4f4d-888a-8db0a89e0e5c"
  network_uris.3354844901: "/rest/ethernet-networks/736ce75d-3c0d-4567-9155-72c112a0a194"
  network_uris.3440930229: "/rest/ethernet-networks/8d0e6475-d7c7-4add-90a8-0f4e3b56b086"
  network_uris.3988505752: "/rest/ethernet-networks/78457b57-b498-401c-b7f9-1e279bc2bd20"
  network_uris.4179382199: "/rest/ethernet-networks/51c9fb6-08c4-43c4-aad6-de59a04e4c63"
  network_uris.77817048: "/rest/ethernet-networks/dd9de825-0d70-4493-865b-f6bd43a70666"
  scopes_uri: <computed>
  state: <computed>
  status: <computed>
  type: "network-setV5"
  uri: <computed>

Plan: 3 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-network$
```

- Now that we have evaluated the plan, we can run an apply to perform all the actions:

```
terraform apply
```

- When prompted, type **Yes** to apply the actions shown:

```

network_uris.3440950229:
network_uris.3988505752:
network_uris.4179382199:
network_uris.77817048:
scopes_uri:
state:
status:
type:
uri:

/rest/ethernet-networks/800e0475-07c7-4aud-90a8-014e30560080
"/rest/ethernet-networks/78457b57-b498-401c-b7f9-1e279bc2bd20"
"/rest/ethernet-networks/51c9fbf6-08c4-43c4-aad6-de59a04e4c63"
"/rest/ethernet-networks/dd9de825-0d70-4493-865b-f6bd43a70666"
<computed>
<computed>
<computed>
"network-setv5"
<computed>

Plan: 3 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: 

```

- Open the OneView UI (<https://192.168.56.101/#/profiles>) to show the progress of the tasks, start with the network page to show the creation of the new network 'Prod_60', then move to the Logical Interconnect Group 'LIG-FlexFabric' page to show the new network added to the Uplink Set 'Prod', then move to the Logical Interconnect 'LE-Synergy-Local-LIG-FlexFabric' that should be detected as inconsistent but eventually updated from the new LIG definition and then finally show the network set with the new network.

Name	VLAN	Type
Deployment	1500	Ethernet
ESX Mgmt	1131	Ethernet
ESX vMotion	1132	Ethernet
Mgmt	100	Ethernet
Prod_60	60	Ethernet
Prod_1101	1101	Ethernet
Prod_1102	1102	Ethernet
Prod_1103	1103	Ethernet

- Once completed, go back to the VS Code console to show the result of the Terraform configuration:

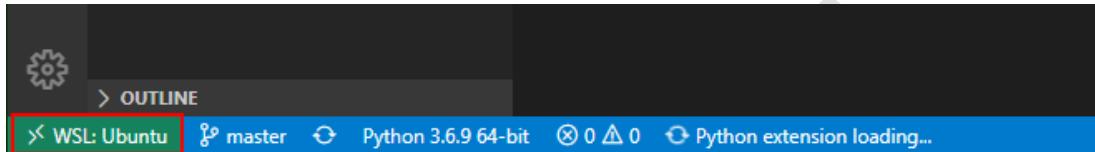
This concludes Terraform – Scenario 1

Terraform – Scenario 2 – Provisioning New Servers

In this scenario, we are going to use Terraform to automate the creation of a server profile using an existing Server Profile Template.

Prerequisites:

- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



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

To automate the creation of a server profile using Terraform, we need to create 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 the following Terraform configuration file using:

```
touch create-profile.tf
cat > create-profile.tf << 'EOF'

provider "oneview" {
  ov_domain = "Local"
  ov_username = "Administrator"
  ov_password = "password"
  ov_endpoint = "https://192.168.56.101"
  ov_sslverify = false
  ov_apiversion = 1600
  ov_ifmatch = "*"
}

// Get Server Hardware
data "oneview_server_hardware" "server_hardware" {
  name = "Synergy-Encl-1, bay 3"
}

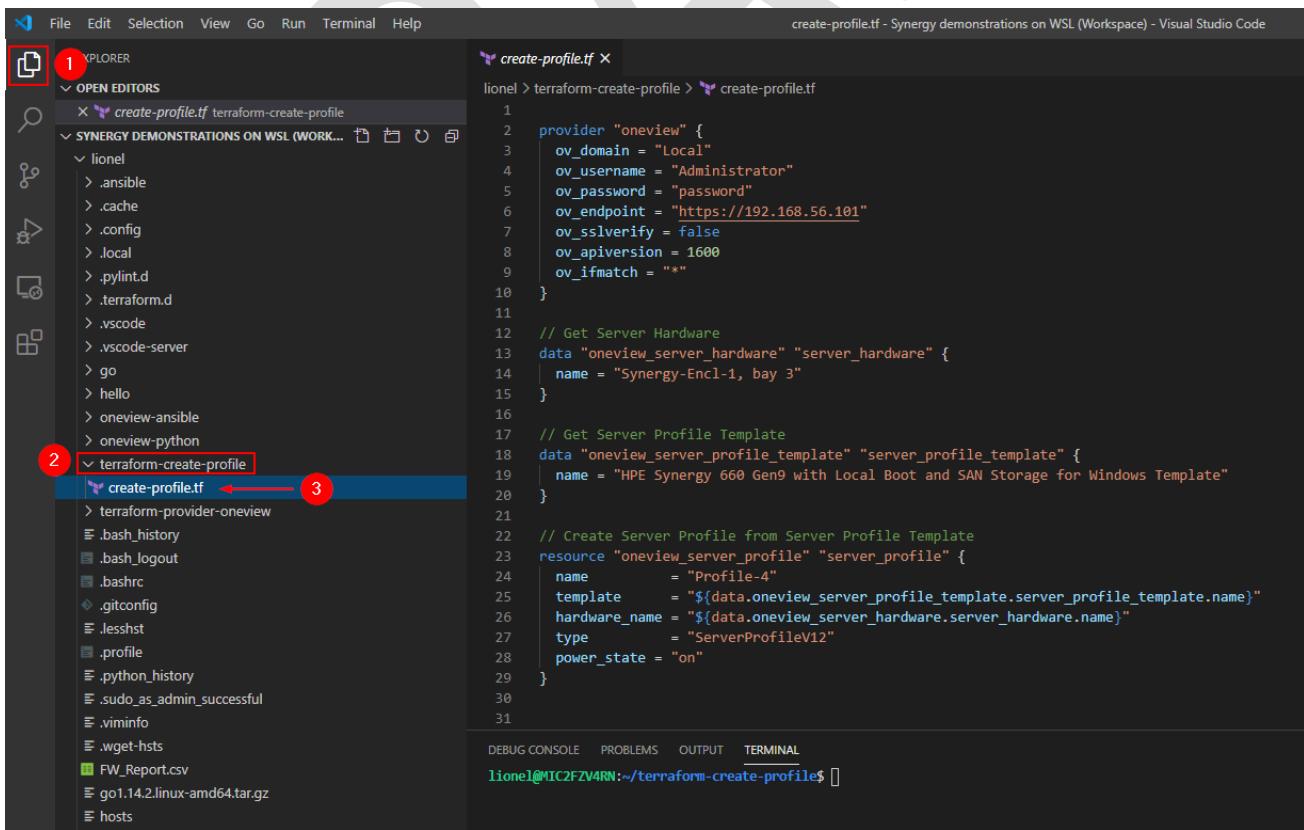
// Get Server Profile Template
data "oneview_server_profile_template" "server_profile_template" {
  name = "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
}

// Create Server Profile from Server Profile Template
resource "oneview_server_profile" "server_profile" {
  name          = "Profile-4"
  template      = "${data.oneview_server_profile_template.server_profile_template.name}"
  hardware_name = "${data.oneview_server_hardware.server_hardware.name}"
  type          = "ServerProfileV12"
  power_state   = "on"
}
EOF
```

Technical white paper

```
lionel@MIC2FZV4RN:~/terraform-create-profile
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$ cat > create-profile.tf << 'EOF'
>
> provider "oneview" {
>   ov_domain = "Local"
>   ov_username = "Administrator"
>   ov_password = "password"
>   ov_endpoint = "https://192.168.56.101"
>   ov_sslverify = false
>   ov_apiversion = 1600
>   ov_ifmatch = "*"
> }
>
> // Get Server Hardware
> data "oneview_server_hardware" "server_hardware" {
>   name = "Synergy-Encl-1, bay 3"
> }
>
> // Get Server Profile Template
> data "oneview_server_profile_template" "server_profile_template" {
>   name = "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
> }
>
> // Create Server Profile from Server Profile Template
> resource "oneview_server_profile" "server_profile" {
>   name           = "Profile-4"
>   template      = "${data.oneview_server_profile_template.server_profile_template.name}"
>   hardware_name = "${data.oneview_server_hardware.server_hardware.name}"
>   type          = "ServerProfileV12"
>   power_state   = "on"
> }
>
> EOF
lionel@MIC2FZV4RN:~/terraform-create-profile$
```

- After the creation of the configuration files in our Terraform project, you can move back to the VS Code console and open the new Terraform file:



- From the VS Code console, go to the *terraform-create-profile* folder:

```
cd ~/terraform-create-profile
```

- Now we need to run the *init* command which will prepare Terraform in this directory:

```
terraform init
```



```
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.

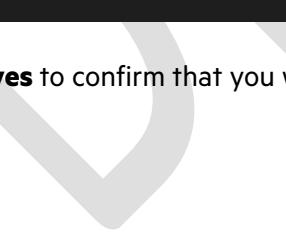
-----
Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

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

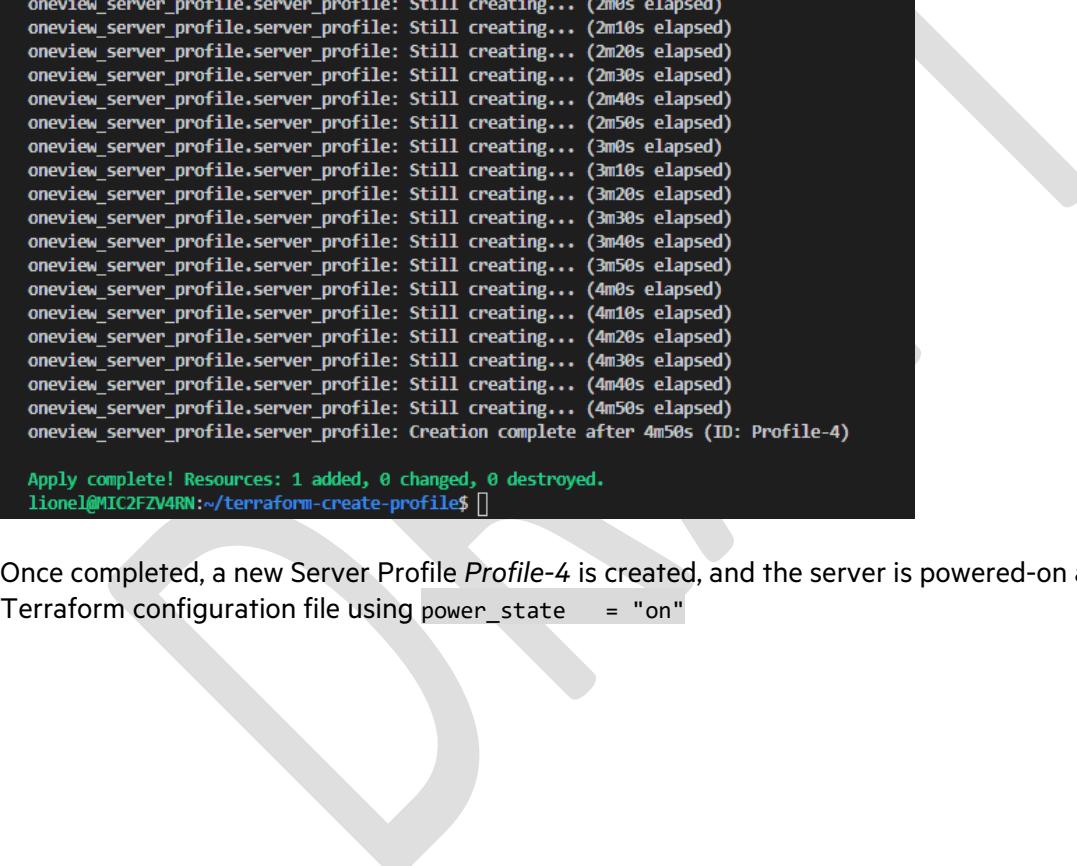
oneview_server_profile.server_profile: Creating...
affinity:      "" => "Bay"
hardware_name: "" => "Synergy-Encl-1, bay 3"
hardware_uri:  "" => "<computed>"
hide_unused_flex_nics: "" => "true"
ilo_ip:        "" => "<computed>"
mac_type:      "" => "Virtual"
name:          "" => "Profile-4"
power_state:   "" => "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 4m50s (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 panel shows 'Profile-4' details under 'General'. It includes fields like Description (Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows), Server hardware (SY 660 Gen9.1), and Device bay (On). A large watermark 'DRAFT' is overlaid on the bottom half of the screen.

Name	Value
Description	Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template
Server hardware	Synergy-Encl-1_bay_3
Server hardware type	SY 660 Gen9.1
Enclosure group	EG-Synergy-Local
Affinity	Device bay
Server power	On
Serial number (v)	VCG0NC300D
UUID (v)	c5e2da8e-3d6f-497e-abf1-cff9b2661ee1
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg0nc300d

This concludes Terraform – Scenario 2

Terraform – Scenario 3 – Unprovisioning Running Server

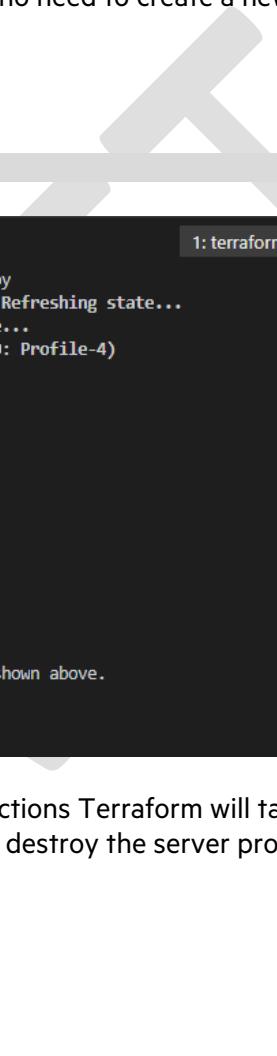
Now that the server that has been created in scenario 2 is up and running, we can explain to the customer that as easily as the provisioning has been made, we can now unprovision the server and return the server hardware back to the OneView resource pool.

Note: This scenario can only be run 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:

Server Profiles 6

Profile-4 General

● Create Completed 4m18s

Name
Profile-1
Profile-2
Profile-3
Profile-4
RH-1
RH-2

General

Description: Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
[HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template](#)

Server hardware: [Synergy-Encl-1.bay.3](#)

Server hardware type: [SY 660 Gen9.1](#)

Enclosure group: [EG-Synergy-Local](#)

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:

Server Profiles 6

Profile-4 General

○ Delete Clearing SAN configuration.

Name
Profile-1
Profile-2
Profile-3
Profile-4
RH-1
RH-2

General

Description: Server Profile Template for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
[HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template](#)

Server hardware: [Synergy-Encl-1.bay.3](#)

Server hardware type: [SY 660 Gen9.1](#)

Enclosure group: [EG-Synergy-Local](#)

Affinity: Device bay

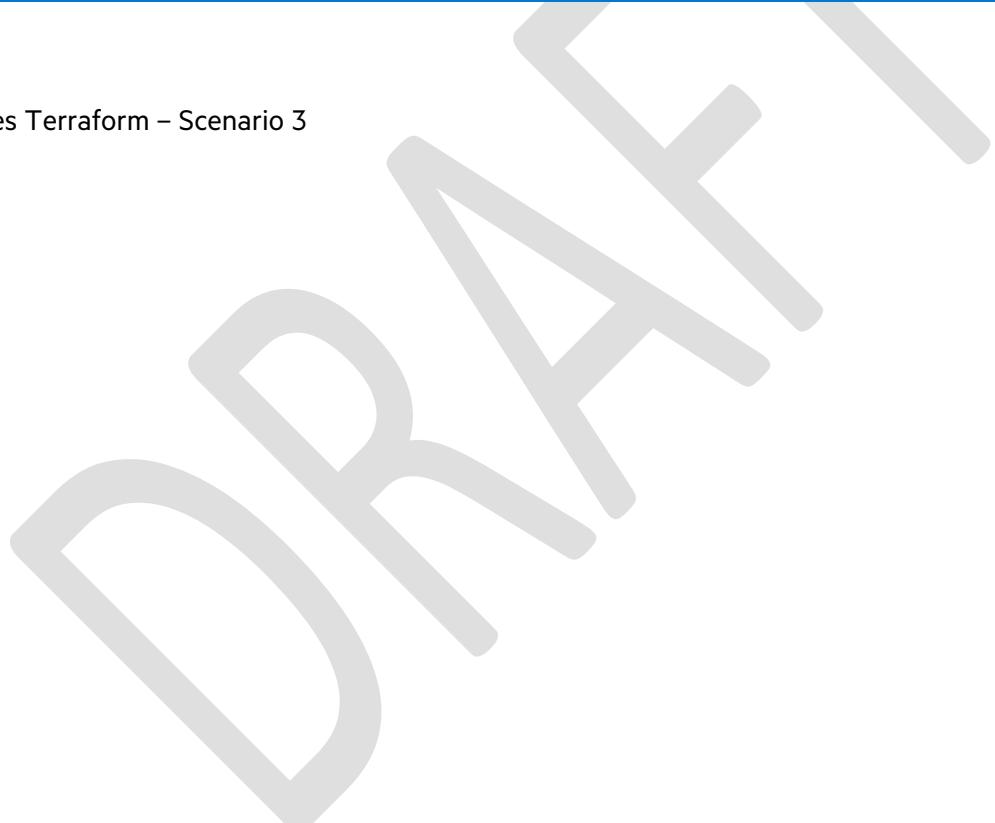
Server power: Off

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:



```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
1: bash + □ ×

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.
lionel@MIC2FZV4RN:~/terraform-create-profile$
```

This concludes Terraform – Scenario 3

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.

DRAFT



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:

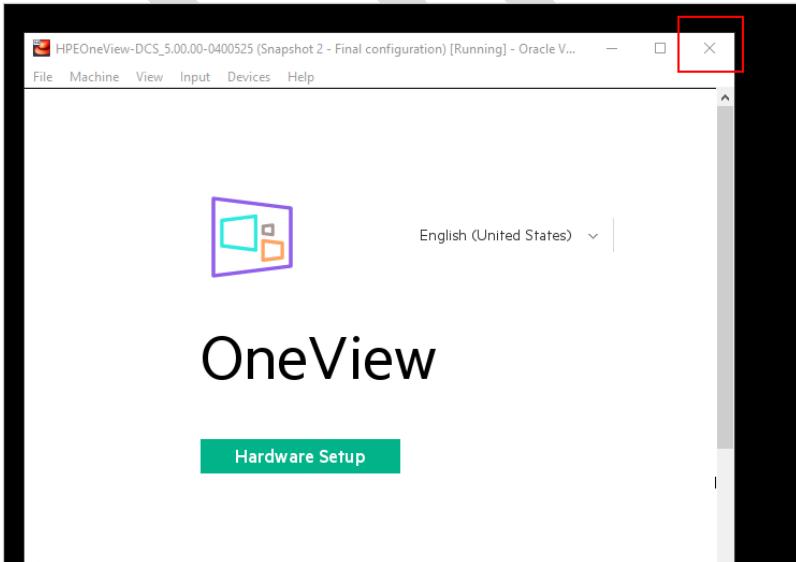
- 1- First snapshot: OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network)
 - ⇒ Can be used to show how to automate the setup of Synergy and the power of our infrastructure 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

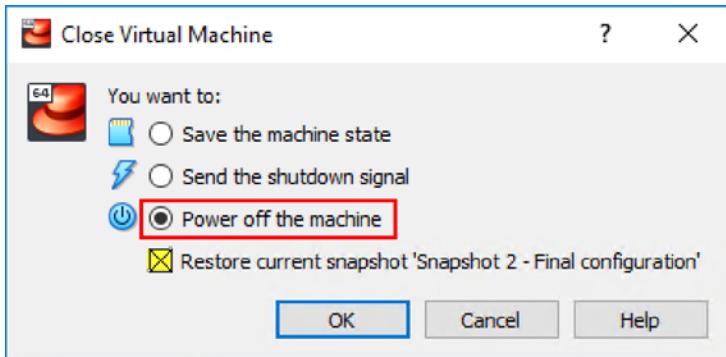
With VirtualBox, when shutting down a virtual machine, you have a "Restore current snapshot" option if you want VirtualBox to load the most recent snapshot the next time you start up the virtual machine again. This is a very convenient and easy option to restore the appliance back to the pre-configured 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.

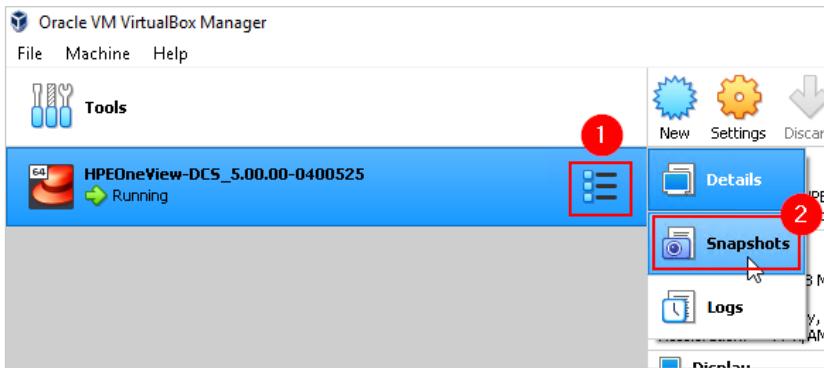
Note: By default, VirtualBox shutdown dialog provides a restore snapshot option with the latest created snapshot.

Now the next time you start the DCS appliance, you will be ready to run your live demonstration scenarios again.

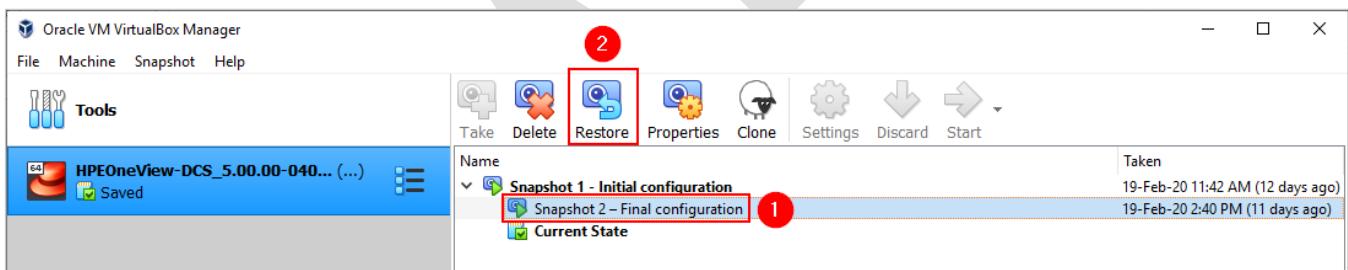
Resetting the Synergy Composer Demonstration appliance to the fully configured state

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.

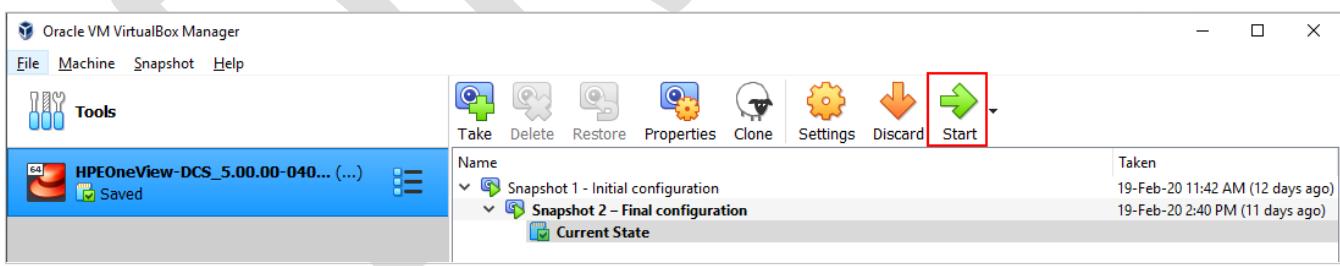
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- 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.

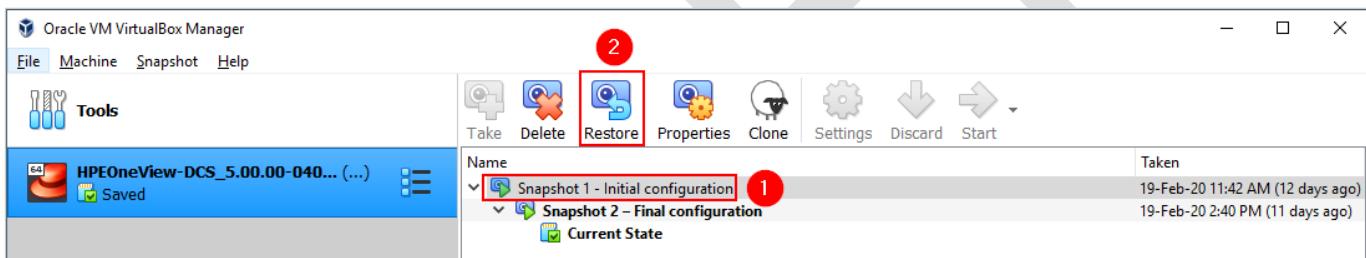
Resetting the Synergy Composer Demonstration appliance to the unconfigured state

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.

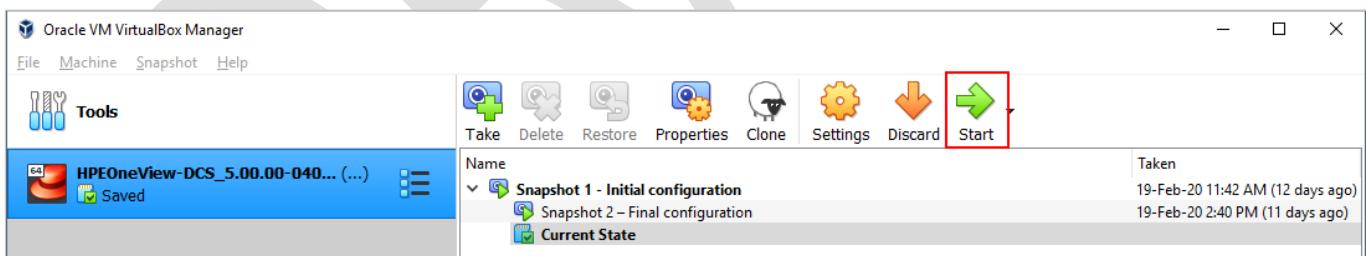
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- 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 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 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
rm -r -f oneview-python
git clone https://github.com/HewlettPackard/oneview-python.git
cd oneview-python
```

```
python3 setup.py install --user
```

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
```

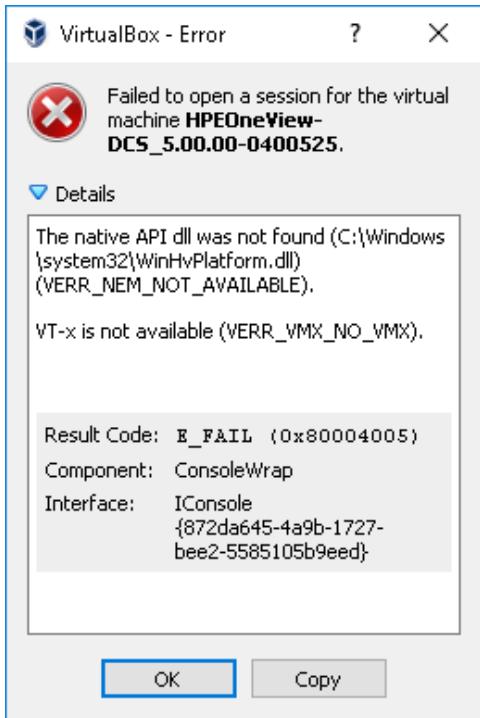
DRAFT



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

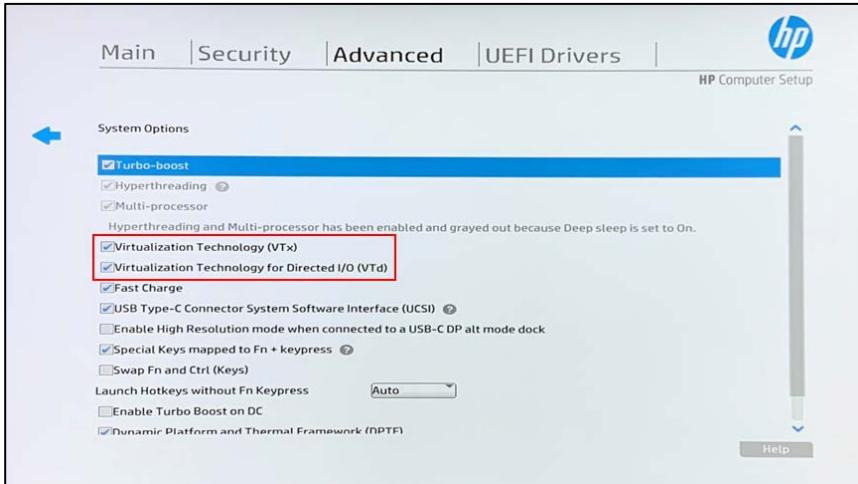


Intel VT-x is a set of processor enhancements to improve virtualization performance.

Oracle VM VirtualBox uses hardware virtualization and requires Intel VT-X to be enabled or not exclusively used by other software.

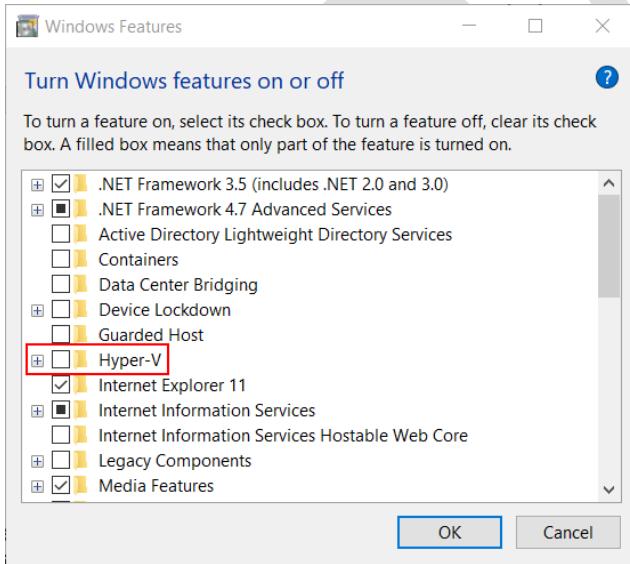
Solution:

- 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)

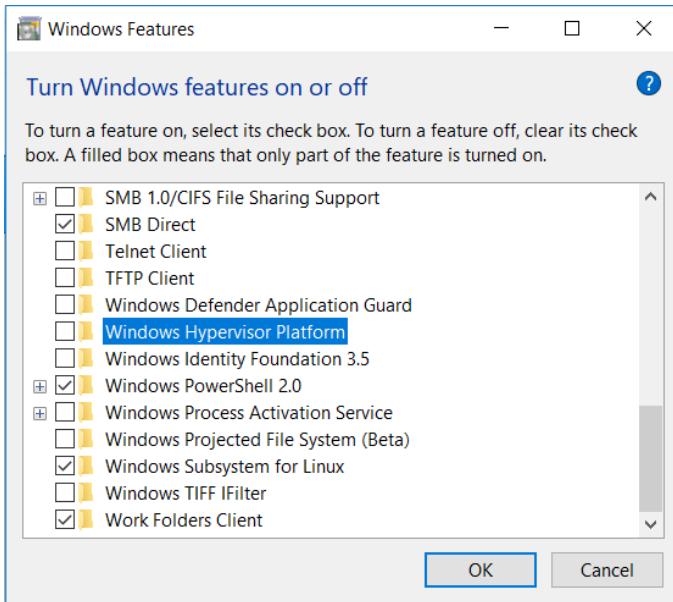


You can also run the following command to disable Hyper-V:

```
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).

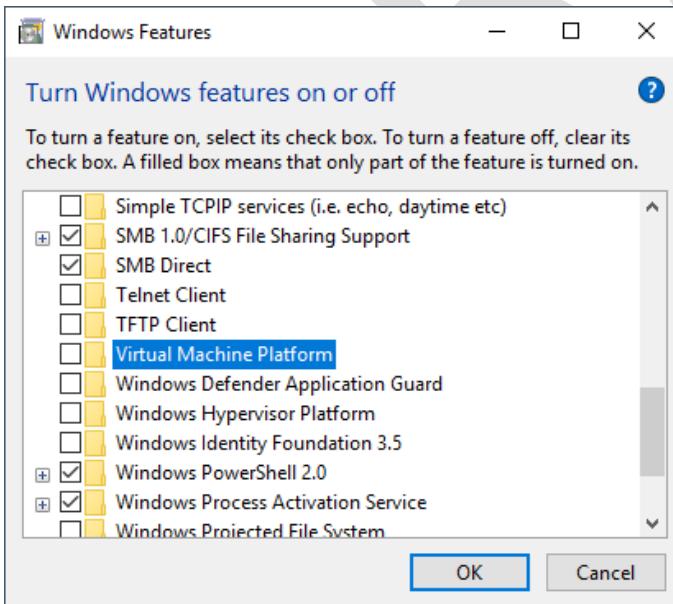


Or run:

```
dism.exe /Online /Disable-Feature:HypervisorPlatform
```

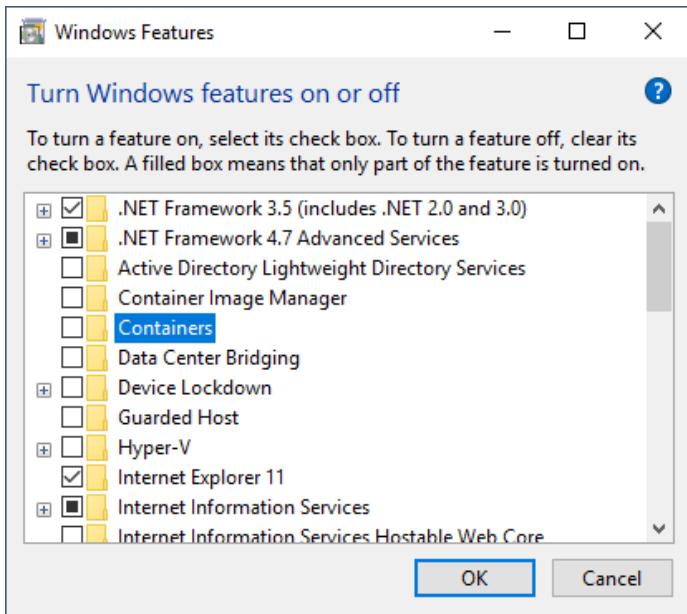
Note: If running, *Windows Hypervisor Platform* is taking exclusive use of VT-x so it is required to turn it off

- Make sure *Virtual Machine Platform* is disabled from Windows features (if present).



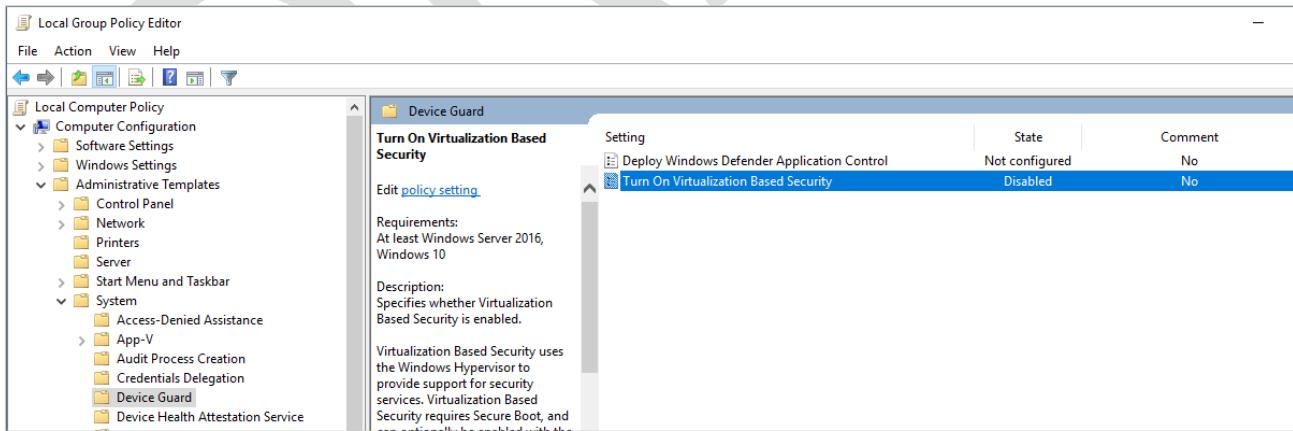
Note: If running, *Virtual Machine Platform* is taking exclusive use of VT-x so it is required to turn it off

- Make sure *Container* is disabled from Windows features (if present).



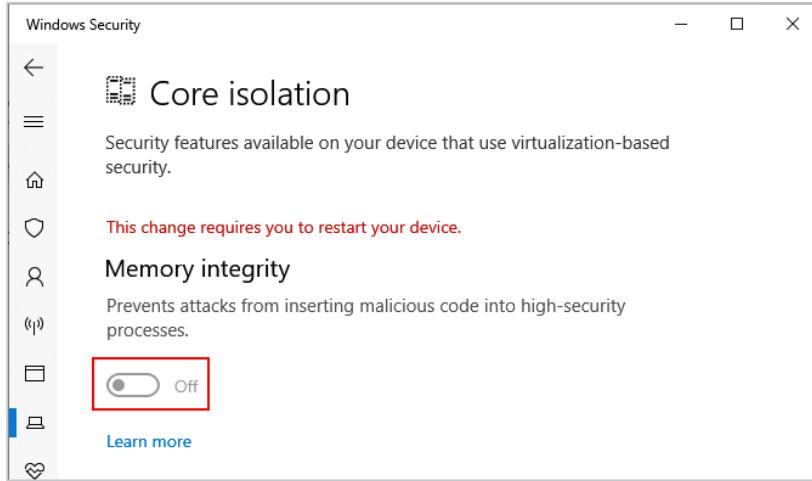
Note: If running, *Container* is taking exclusive use of VT-x so it is required to turn it off

- 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.

DRAFT



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.

