



**Hewlett Packard  
Enterprise**

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

Includes Postman, PowerShell, Python and Ansible  
live demonstration scenarios

## Contents

Abstract .....	6
Requirements .....	6
Demonstration appliance prerequisites .....	7
Chapter-1 - Deploying the HPE OneView Demonstration appliance on VirtualBox .....	8
Downloading HPE OneView demonstration appliance.....	8
Downloading CentOS 7.5 Boot ISO .....	11
Downloading and installing VirtualBox on your PC .....	11
Importing and tuning the HPE OneView Demonstration appliance in VirtualBox .....	15
Chapter-2 – Preparing Ubuntu on Windows Subsystem for Linux (WSL) .....	28
Installing Windows Subsystem for Linux (WSL) .....	28
Installing Ubuntu Windows Subsystem for Linux.....	30
Installing Ansible on Ubuntu WSL .....	33
Installing HPE OneView Python SDK on Ubuntu WSL .....	35
Installing Ansible Modules for HPE OneView on Ubuntu WSL .....	37
Installing Git for VS Code source control.....	41
Where do my files live in my Ubuntu WSL? .....	45
Chapter-3 – Preparing Microsoft Visual Studio Code .....	47
Installing Visual Studio Code .....	47
Preparing VS Code for Python .....	56
Preparing VS Code for Ansible.....	61
Preparing VS Code for PowerShell .....	62
Installing the PowerShell library for HPE OneView .....	67
Chapter-4 –Initial Configuration of the Demonstration Appliance .....	69
Hardware discovery and initial appliance configuration.....	69
Creation of an initial setup snapshot .....	74
Chapter-5 – Preparing Postman .....	77
Installing and configuring Postman.....	77
Importing collections .....	78
Configuration of the Postman environment .....	81
Chapter-6 - Preparing your demonstration appliance for demos .....	86
Final appliance configuration .....	87
Creation of a final setup snapshot .....	99
Chapter-7 – Preparing the live demonstration scenarios.....	102



# Hewlett Packard Enterprise

Demonstrating Synergy Composer and OneView Key features .....	102
Demonstrating Infrastructure programmability .....	102
Postman - Scenario 1 – Introduction to the OneView REST API	103
PowerShell – Scenario 1 - Day-to-day operation task automation	112
PowerShell – Scenario 2 - Creating a report	121
PowerShell – Scenario 3 - Accelerating a configuration change	126
Python – Scenario 1 - Day-to-day operation task automation	136
Python – Scenario 2 – Creating a report	144
Python – Scenario 3 - Accelerating a configuration change	148
Ansible – Scenario 1 – Collecting facts in OneView	154
Ansible – Scenario 2 – Provisioning New Servers	165
Ansible – Scenario 3 – Unprovisioning Running Servers	177
Chapter-8 - How to reset the Synergy demonstration environment.....	182
Shutting down the Synergy Composer Demonstration appliance .....	182
Resetting the Synergy Composer Demonstration appliance to the fully configured state .....	184
Resetting the Synergy Composer Demonstration appliance to the unconfigured state.....	185
Troubleshooting .....	186
Hardware and Software versions used to build this demonstration environment.....	191
Summary .....	192



# Hewlett Packard Enterprise

## Abstract

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

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

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

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

Live demonstration scenarios include:

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

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

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

## Requirements

- A powerful laptop with minimum 16G of RAM (best is 32G) and with Intel Virtualization Technology enabled
- CentOS 7.5 Boot ISO
- OneView DCS 5.00 OVA File(s) for Synergy or for BL/DL, the process below is 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

For HPE Employees:

- Log in using your HPE Passport credentials to *HPE My Enterprise License portal*  
<https://myenterprisicense.hpe.com/cwp-ui/auth/login>
- Select **Software for HPE Employees**
- Select **OneView** in the Software Category drop down menu
- Scroll down and select **HPE OneView Demonstration Appliance**

The screenshot shows the HPE website's product page for the OneView Demonstration Appliance. At the top, there's a dark header with the HPE logo and navigation links for Solutions, Services, Products, About Us, and Support. Below the header, a large green button labeled "Download" is prominently displayed. To its left, a text box contains a brief description of the appliance. On the right, there's a "Leave Feedback" button. The main title "HPE OneView Demonstration Appliance" is centered above the download button. The overall layout is clean and professional.

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



# Hewlett Packard Enterprise

- Accept the software terms and conditions
- Select **HPE\_OneView\_DCS\_5.00\_Synergy\_ESXi\_Z7550-96681.ova** then click **Download**

The screenshot shows a web browser interface for 'Software Evaluations'. At the top, there's a navigation bar with a home icon, a back arrow labeled 'Software Evaluations', and user profile icons. A prominent green banner at the top states 'ACTIVATION COMPLETE. Thank you for registering.' Below this, the 'Product Family: HPE OneView' is displayed. Underneath, there are two buttons: 'Download Files' (which is underlined) and 'View Activation Details'. The main content area is titled 'Download Files' and contains a section for 'Software (18)'. Three items are listed:

- [HPE\\_ONEVIEW\\_DCS\\_5.00\\_SYNERGY\\_ESXI\\_Z7550-96681.OVA](#) (2.14 GB)
- [HPE\\_ONEVIEW\\_DCS\\_5.00\\_SYNERGY\\_HYPER\\_V\\_Z7550-96682.ZIP](#) (1.86 GB)
- [HPE\\_ONEVIEW\\_DCS\\_5.00\\_SYNERGY\\_KVM\\_Z7550-96683.TAR.GZ](#) (2.08 GB)

To the right of these files is a small grey box containing the text 'Tell me what I can do here...' with a downward arrow, and below it is a link 'Download Files' with the subtext 'The files'.



# Hewlett Packard Enterprise

For HPE Partners:

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

**Partner Ready Portal**

**Passport Sign In**

Email Address

•••••

SIGN IN

New user? Register here

Forgot User ID or Password?

**An easier way to engage**

The HPE Partner Ready Portal delivers easier-to-find, personalised sales tools and resources to provide a faster and more collaborative sales engagement, training, demand generation and

**Want to become an HPE Partner?**

Get started and sign up today

- Select **My Workspace -> Manage Software and Licenses**
- Select **SW evaluations**
- Select **All Categories** and scroll to **OneView**
- Accept the software terms and conditions
- Select **HPE\_OneView\_DCS\_5.00\_Synergy\_ESXi\_Z7550-96681.ova** then click **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](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.

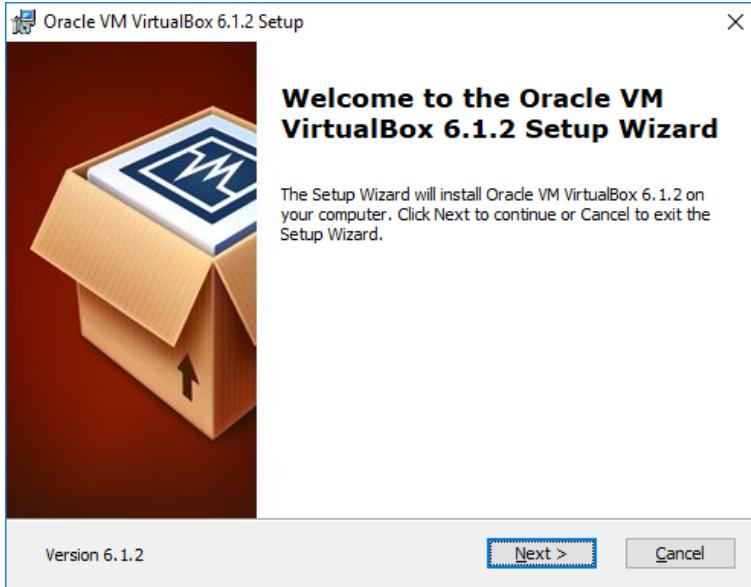
- To download VirtualBox, go to <https://www.virtualbox.org/wiki/Downloads>
- Select **Windows hosts**



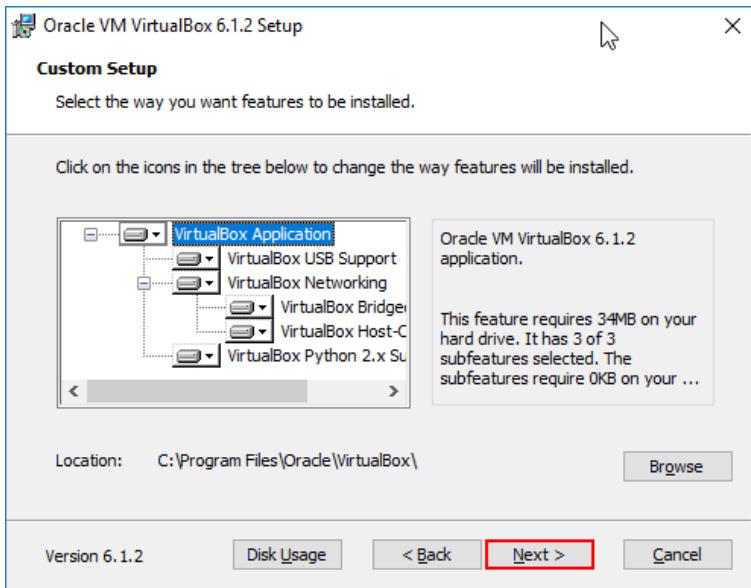


# Hewlett Packard Enterprise

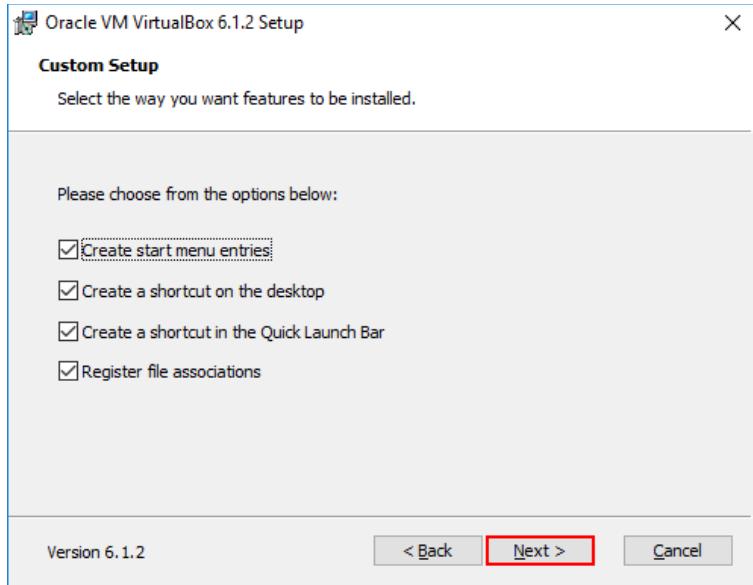
- Save the file on your PC and launch the executable.



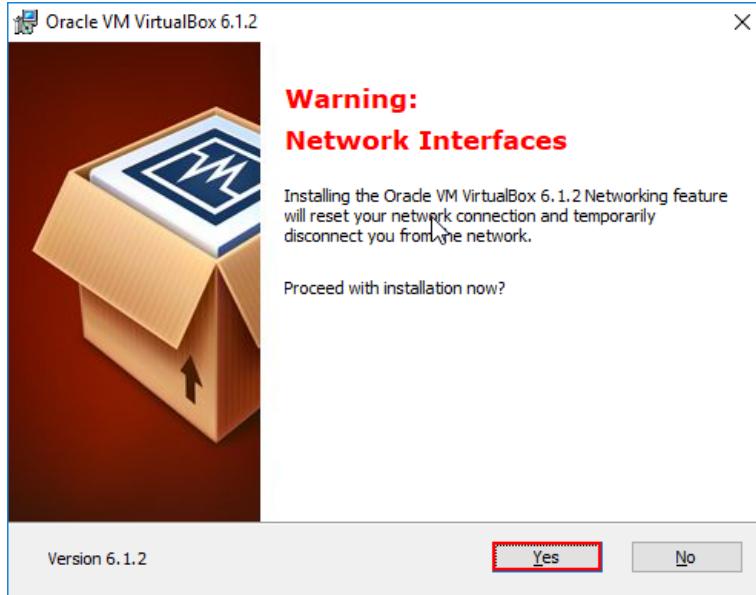
- Leave all default parameters then click **Next**



- Click **Next**



- Click **Next**

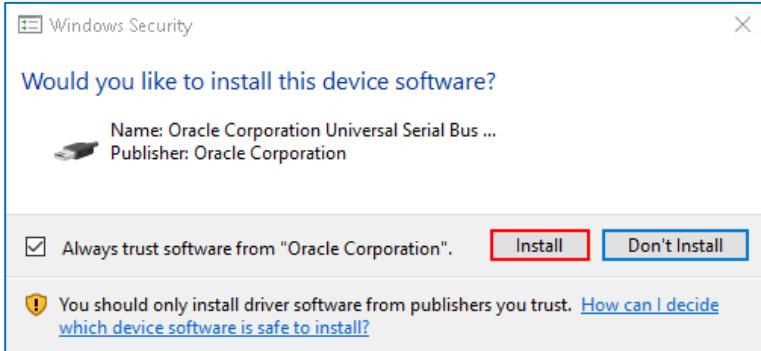


- Then click **Yes** to proceed with the installation then click **Install**



# Hewlett Packard Enterprise

- When the Windows Security message pops-up, click **Install**



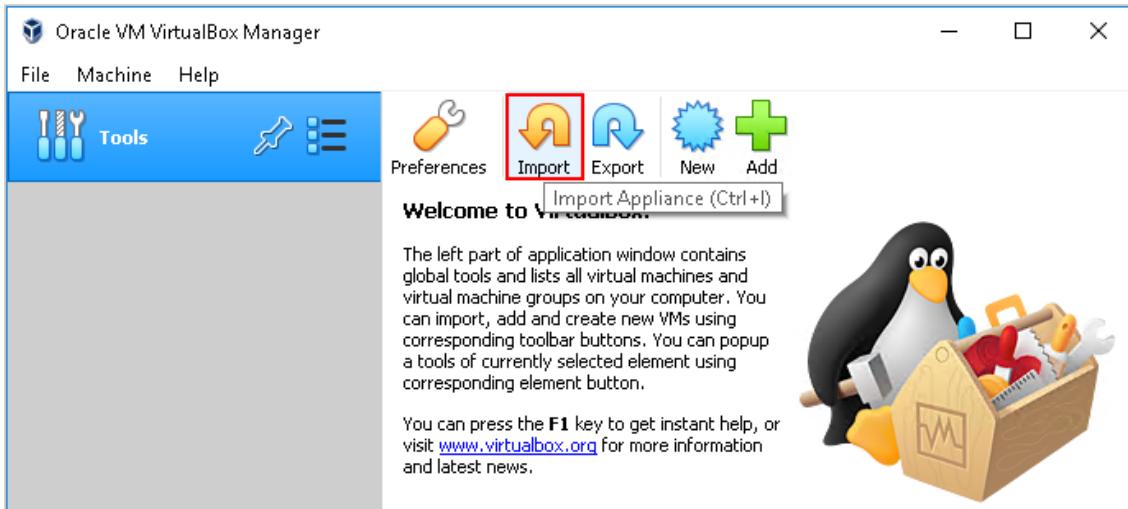
- Then click **Finish** with the start option checked



# Hewlett Packard Enterprise

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

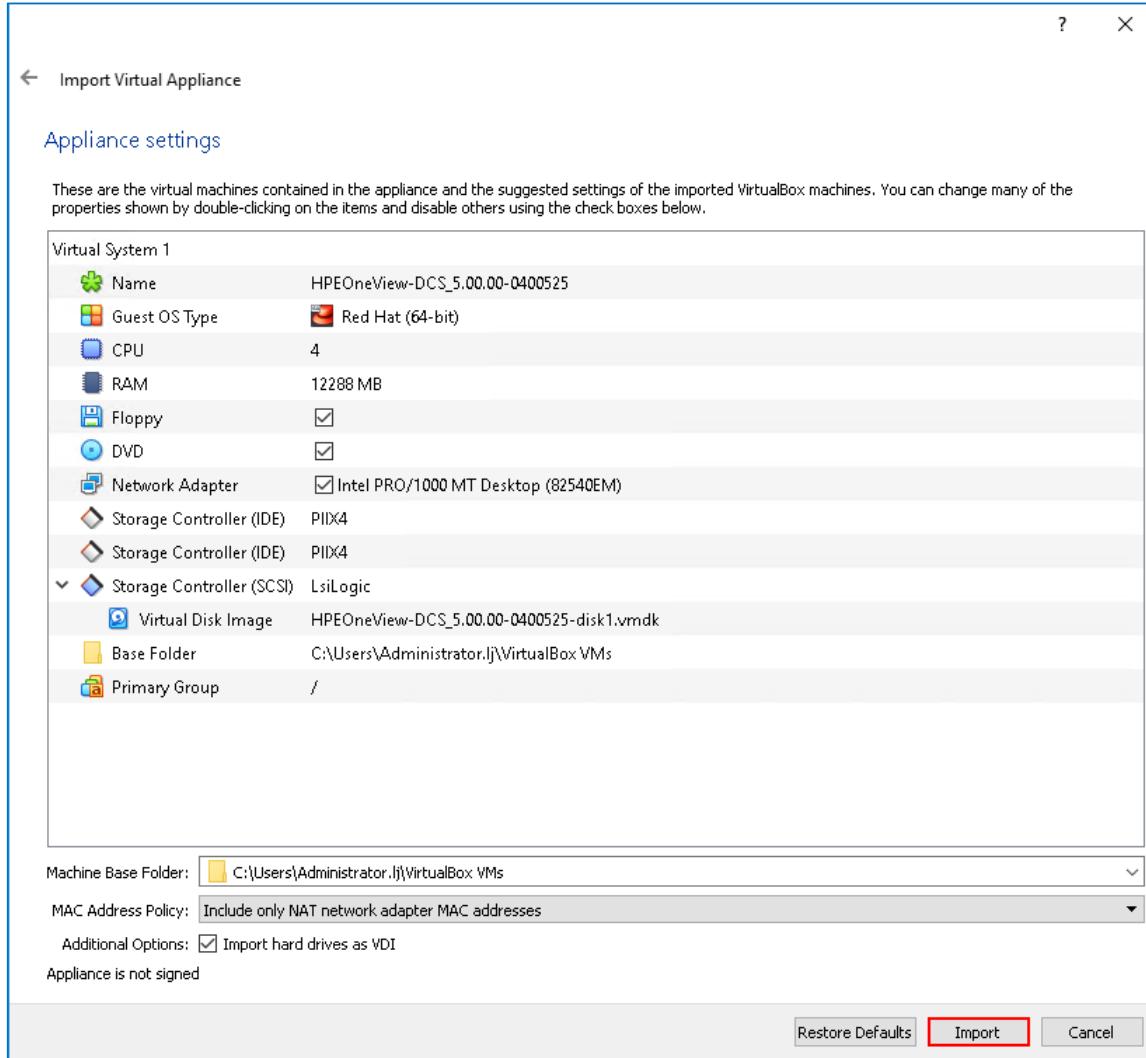
- In VirtualBox, select **Import**



- Select the **HPE\_OneView\_DCS\_5.00\_Synergy\_ESXi\_Z7550-96681.ova** file

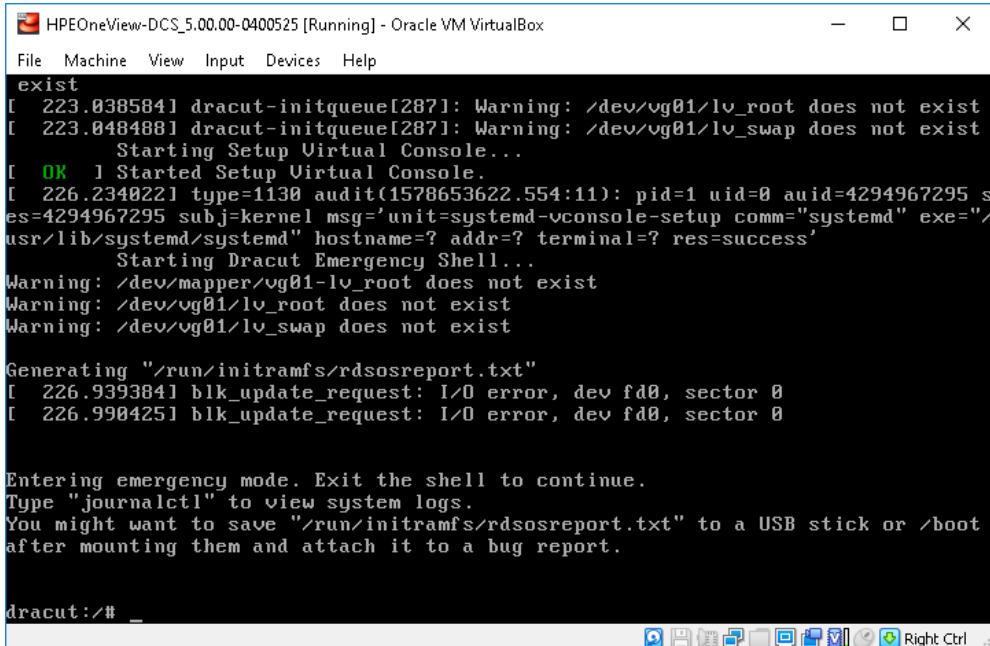
# Hewlett Packard Enterprise

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



# Hewlett Packard Enterprise

**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_5.00.00-0400525 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
exist
[ 223.038584] dracut-initqueue[287]: Warning: /dev/vg01/lv_root does not exist
[ 223.048488] dracut-initqueue[287]: Warning: /dev/vg01/lv_swap does not exist
    Starting Setup Virtual Console...
[ OK ] Started Setup Virtual Console.
[ 226.934022] type=1130 audit(1578653622.554:11): 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

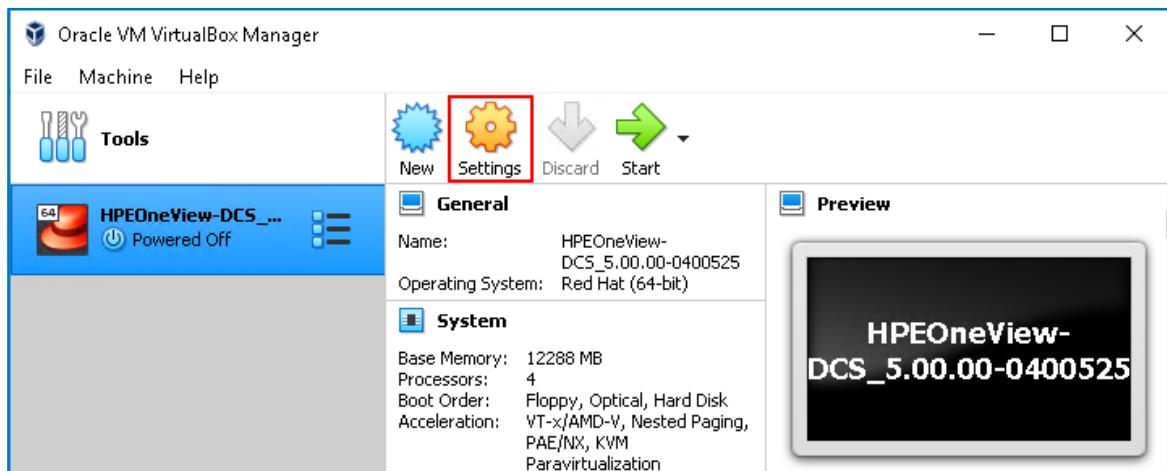
Generating "/run/initramfs/rdsosreport.txt"
[ 226.939384] blk_update_request: I/O error, dev fd0, sector 0
[ 226.990425] blk_update_request: I/O error, dev fd0, sector 0

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:/# _
```

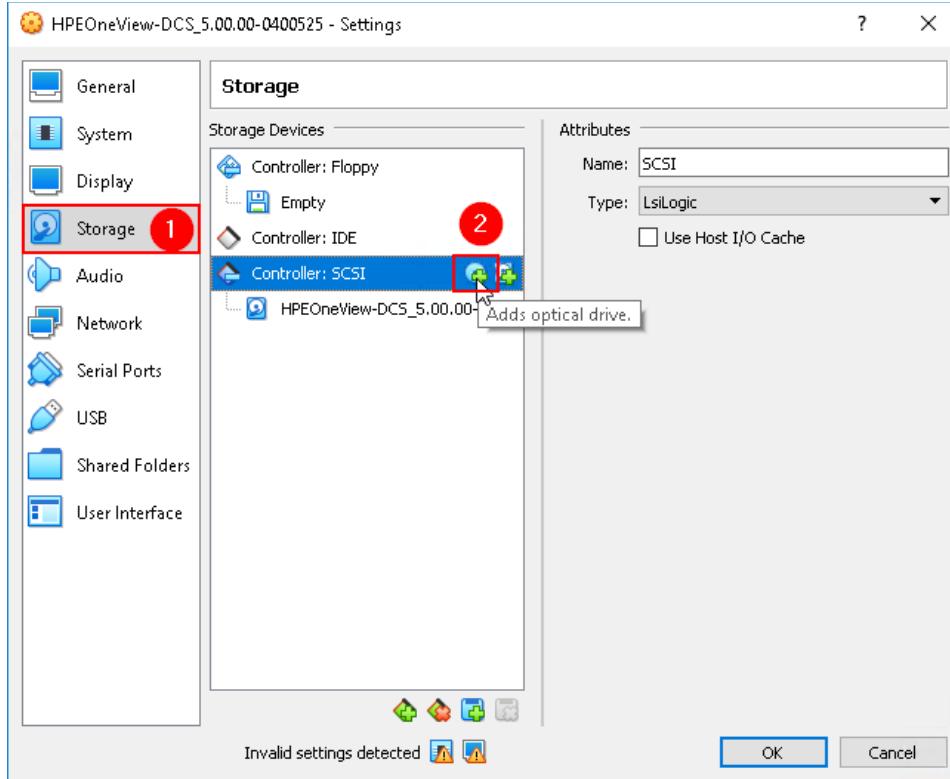
The solution to fix this issue is to inject the missing drivers from a CentOS 7.5 CD ISO.

- Edit the VM Settings

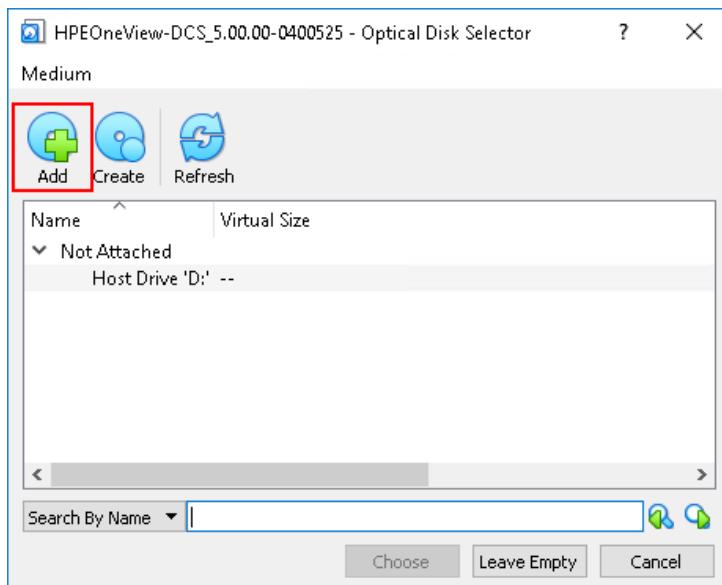


# Hewlett Packard Enterprise

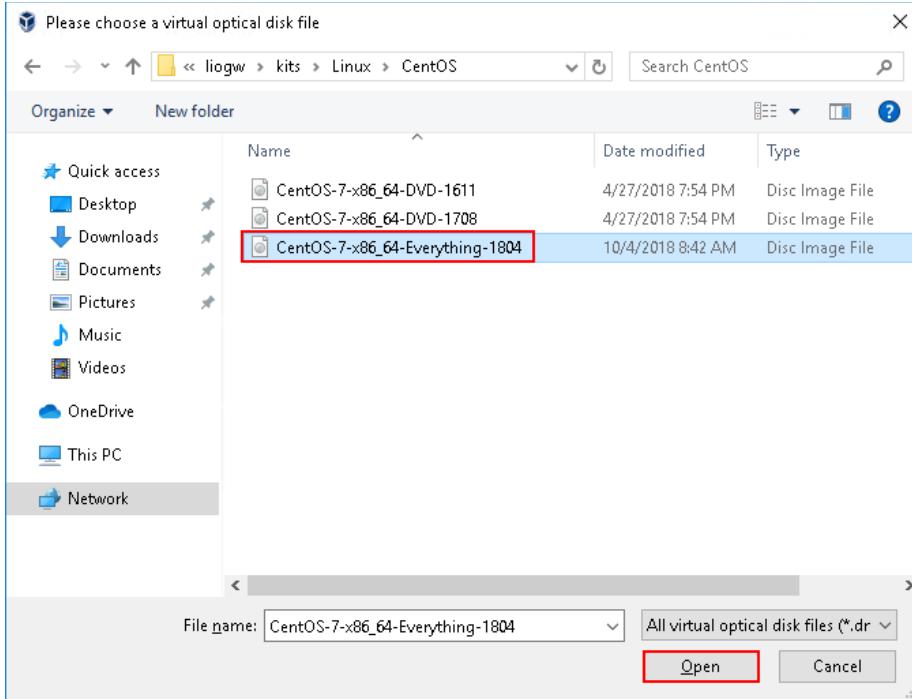
- Select **Storage** then click on **Adds optical drive** icon on the iSCSI Controller



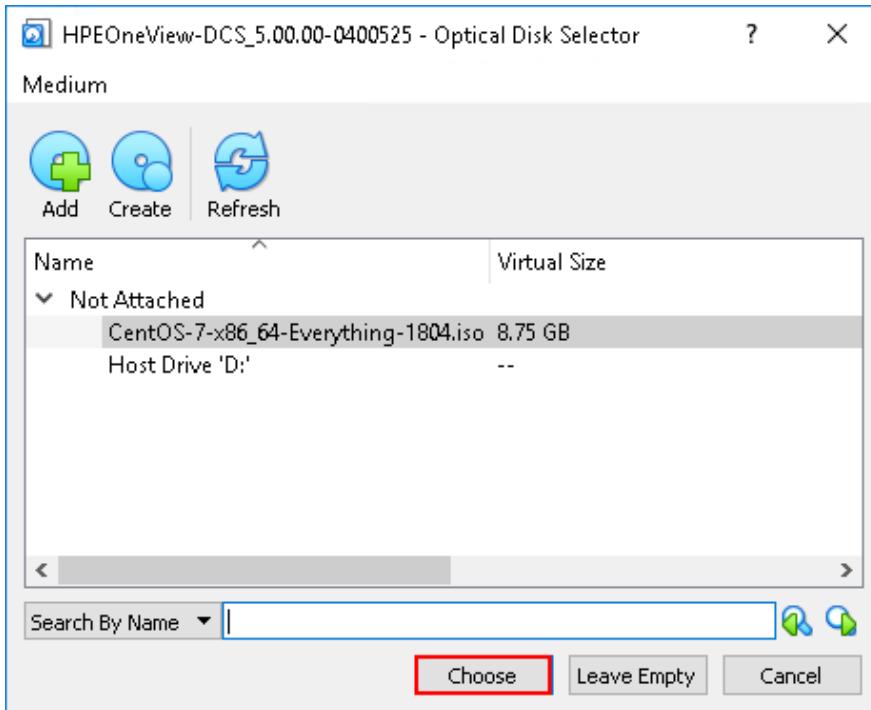
- Select **Add**



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

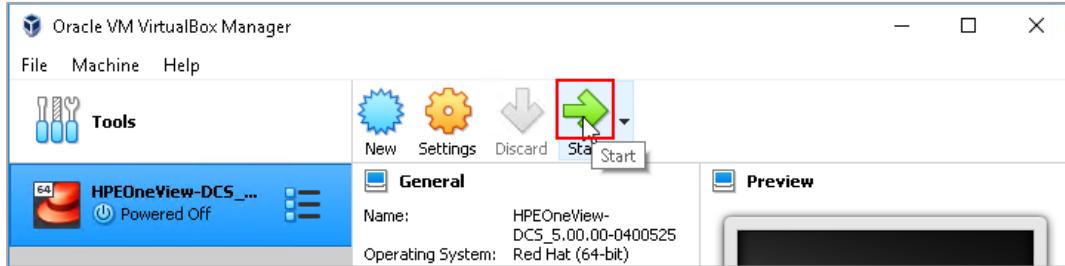


- Click on **Choose** then **OK**



# Hewlett Packard Enterprise

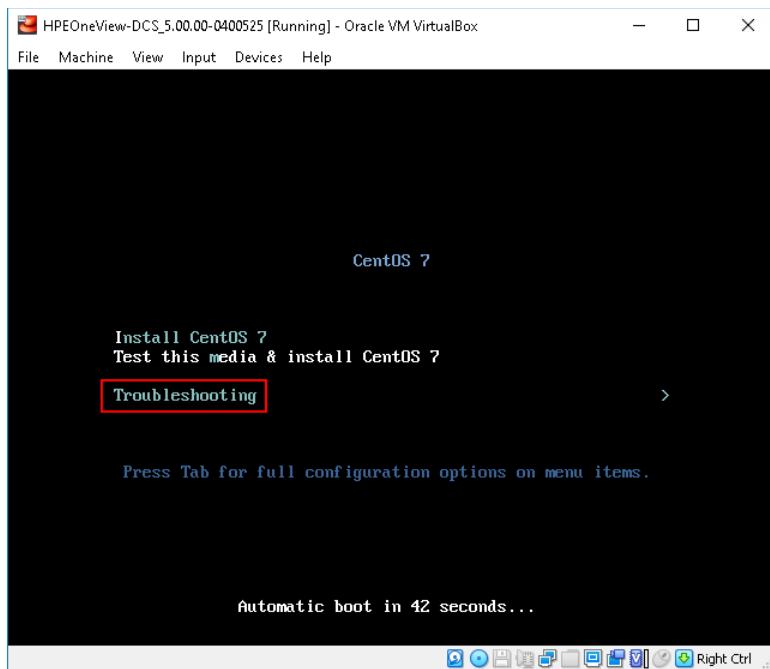
- Then start the VM by pressing **Start**



The CentOS starting menu should come up.

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

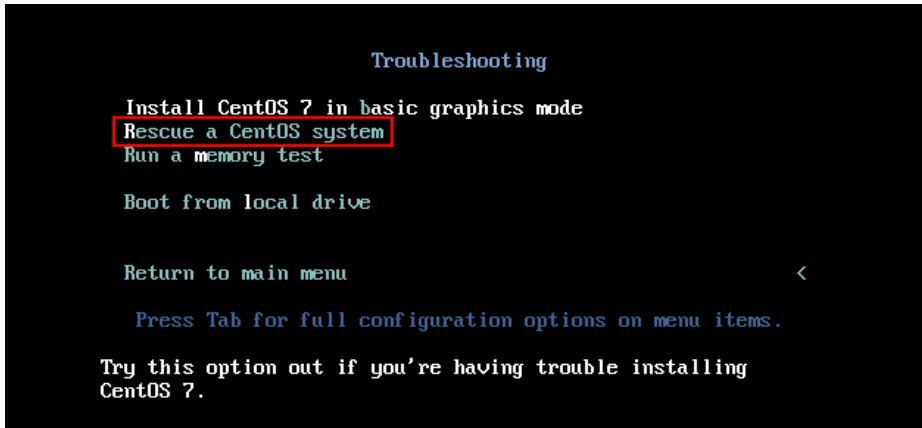
- Select Troubleshooting



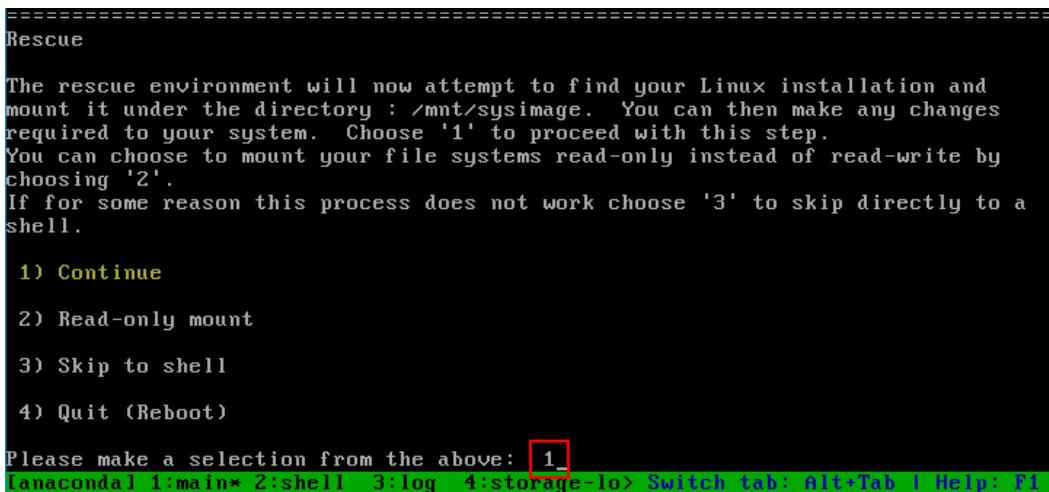


# Hewlett Packard Enterprise

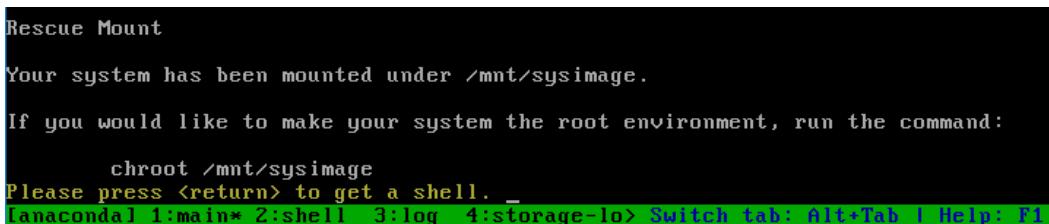
- Then select Rescue a CentOS system



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



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



- 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*

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

- 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 5.00 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 pro-
tected.
#     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 command `exit` and press **enter** to unmount the appliance Image and return to the Rescue Mode shell.

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

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

- Type the following to shut down the VM:

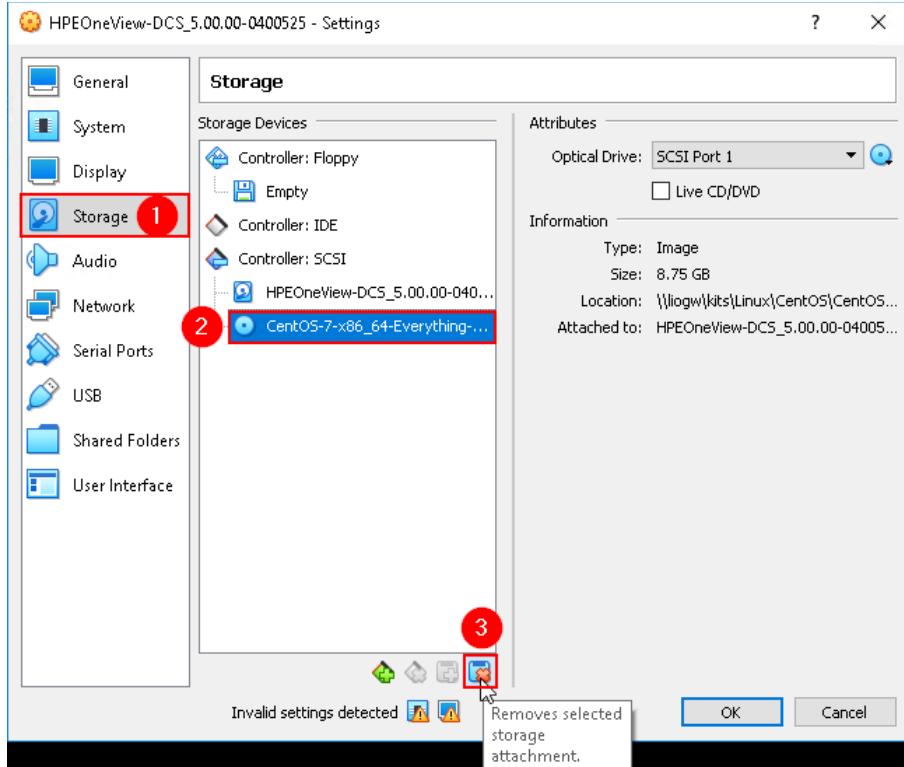
```
shutdown now

[ 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...LGNt6qtij9MViJSQ9Mm0de0TjV8hxJi.
[ 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...
```



# Hewlett Packard Enterprise

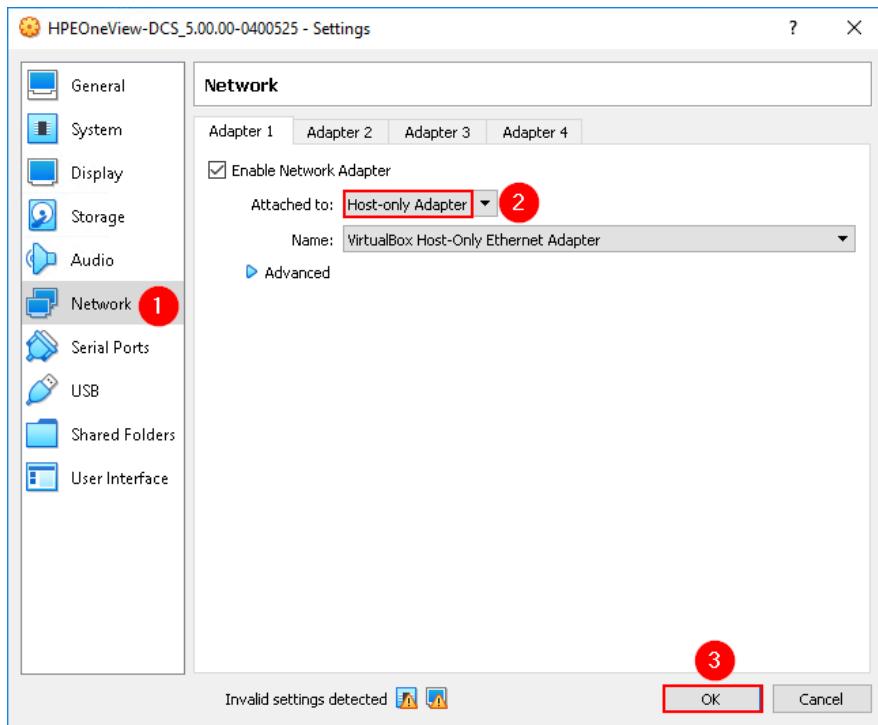
- Once the VM has shutdown, edit the VM **Settings** and remove the CentOS CD.



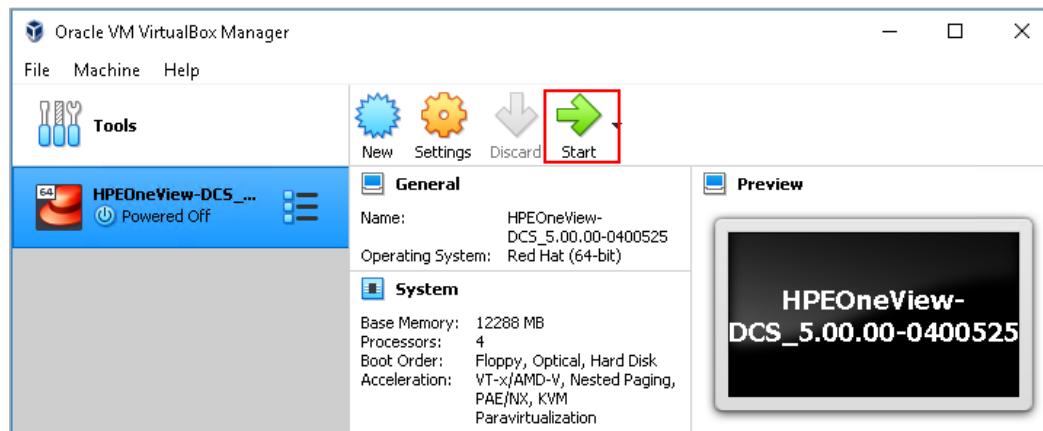


# Hewlett Packard Enterprise

- 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, and then click on **Start** to power on the HPE OneView demonstration appliance VM.



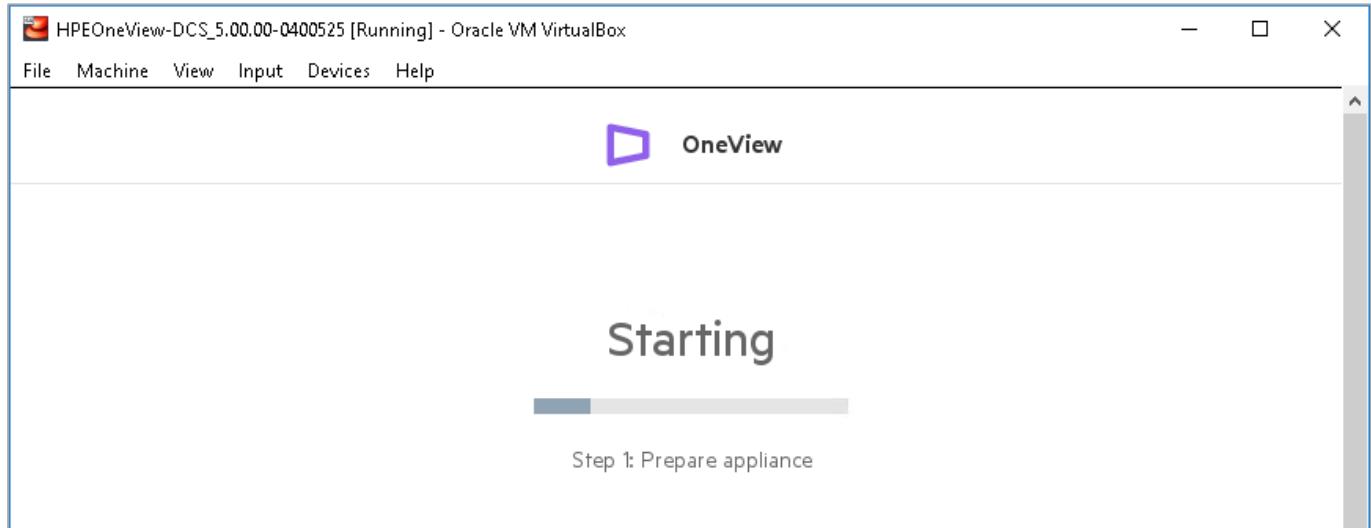


# Hewlett Packard Enterprise

- 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.



# Hewlett Packard Enterprise

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

At this stage you don't need to wait for the boot to complete, continue to the next chapter.

This concludes Chapter-1

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



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

### Installing Windows Subsystem for Linux (WSL)

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

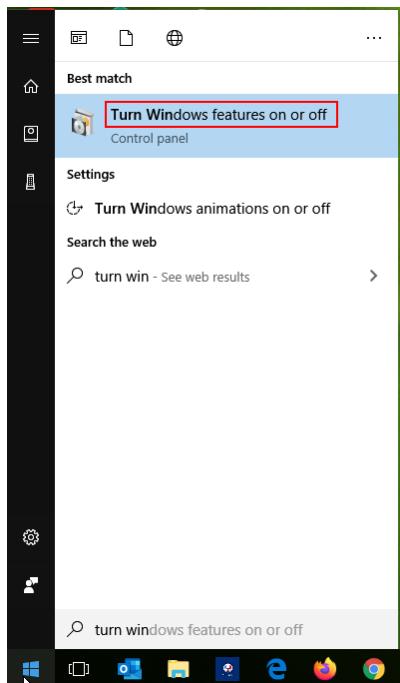
To learn more, see <https://docs.microsoft.com/en-us/windows/wsl>

For our PC demonstration environment, we are going to use WSL to run Python, Ansible and the Ansible module for HPE OneView.

**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 still be used for this lab

- To enable the WSL feature, open the Windows start menu and enter **Turn windows**

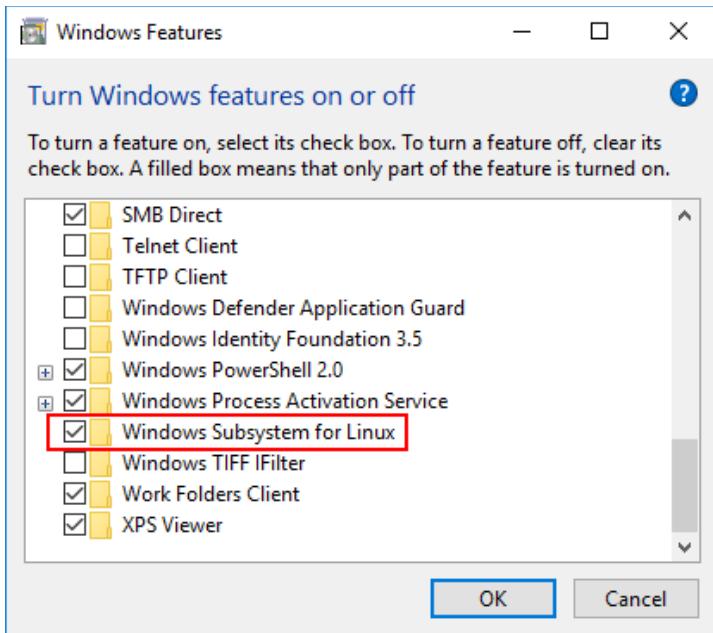


- Then select **Turn Windows Features on and off**



# Hewlett Packard Enterprise

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



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



# Hewlett Packard Enterprise

## 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. Start the Microsoft Store and search for **Linux**.

- Select **Ubuntu**

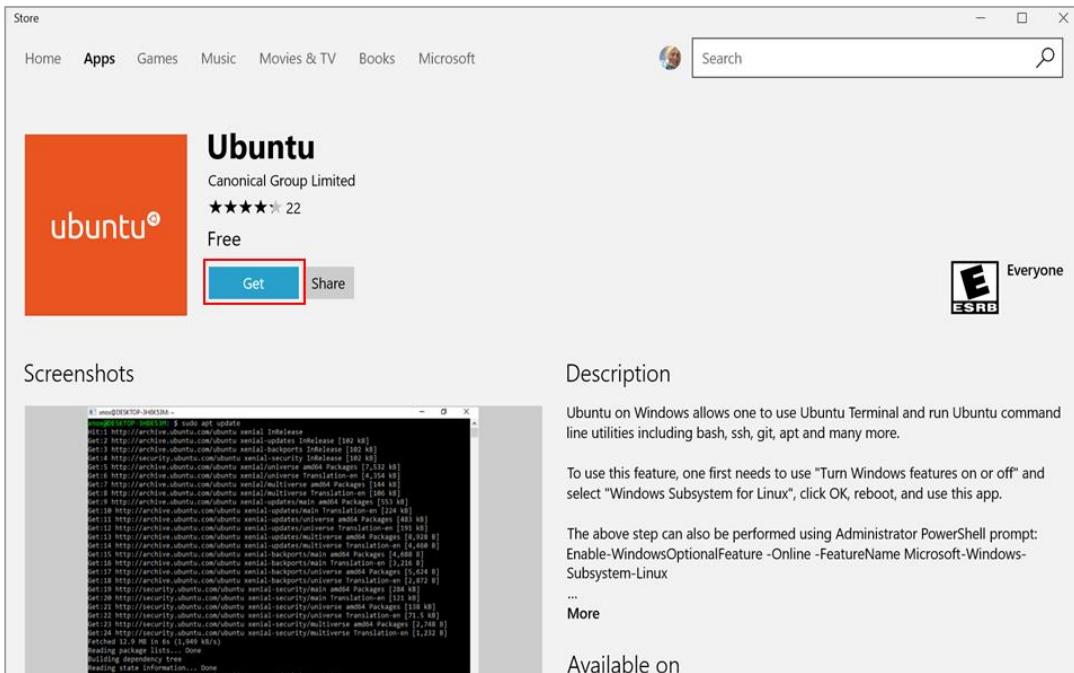
The screenshot shows the Microsoft Store interface. At the top, there is a search bar with the text "Search" and a user profile icon. Below the search bar, the text "Results for: linux" is displayed. There are two dropdown menus: "Departments" set to "All departments" and "Available on" set to "PC". The main area displays a large promotional banner for WSL. The banner features the text "C:\> Linux on Windows? Totally." and "Install and run Linux distributions side-by-side on the Windows Subsystem for Linux (WSL)." It includes an image of a laptop screen showing a terminal window with green text on a black background. A blue button labeled "Get the apps" is visible. Below the banner, the heading "Apps (93) Show all" is shown. A grid of app tiles is displayed, with the first tile, "Ubuntu", highlighted with a red border. Other visible tiles include "Kali Linux BY OFFENSIVE SECURITY", "Ubuntu 18.04 LTS", "Debian", "Linux Cheatsheet", "TeamViewer: Remote Control", and "Raft WSL". Each tile shows the app name, a star rating, and the word "Free".

App	Rating	Downloads	Status
Ubuntu	★★★★★ 254	Free	
Kali Linux BY OFFENSIVE SECURITY	★★★★★ 182	Free	
Ubuntu 18.04 LTS	★★★★★ 173	Free	
Debian	★★★★★ 104	Free	
Linux Cheatsheet	★★★★★ 16	Free*	
TeamViewer: Remote Control	★★★★★ 362	Free	
Raft WSL	★★★★★ 5	Free*	

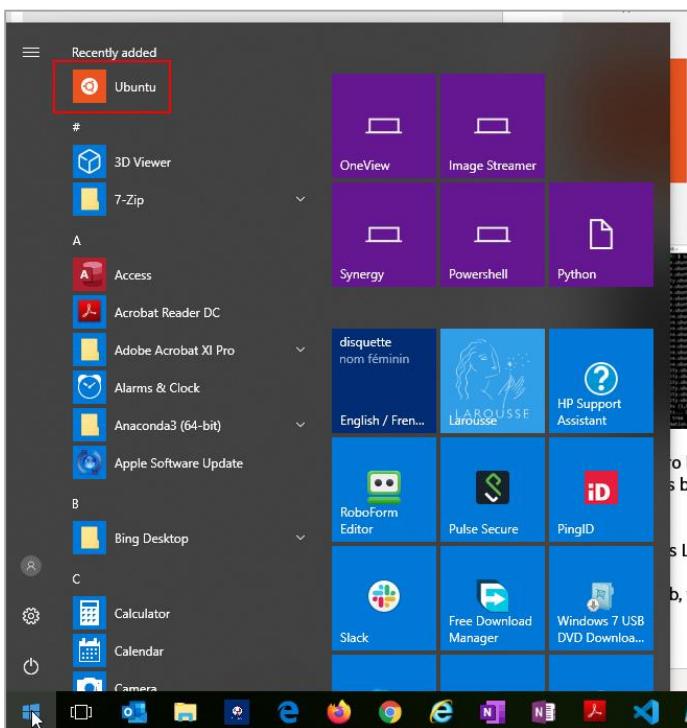


# Hewlett Packard Enterprise

- Then select **Get**

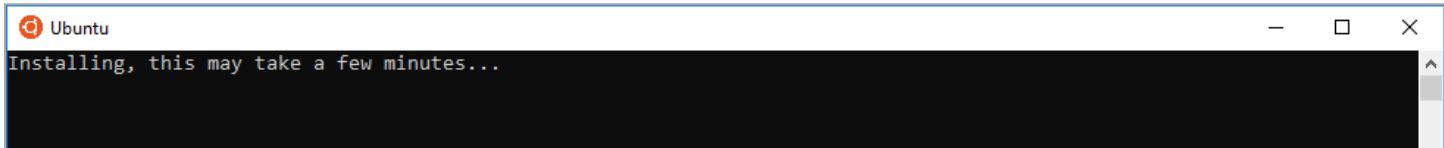


- Once Ubuntu has been downloaded and installed, click the **Launch** button in the Microsoft Store app, or launch Ubuntu from the Start menu;



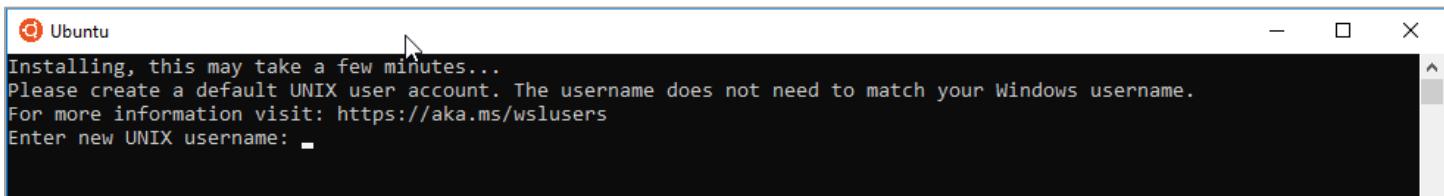
# Hewlett Packard Enterprise

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



**Note:** This may take around a minute or more 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.

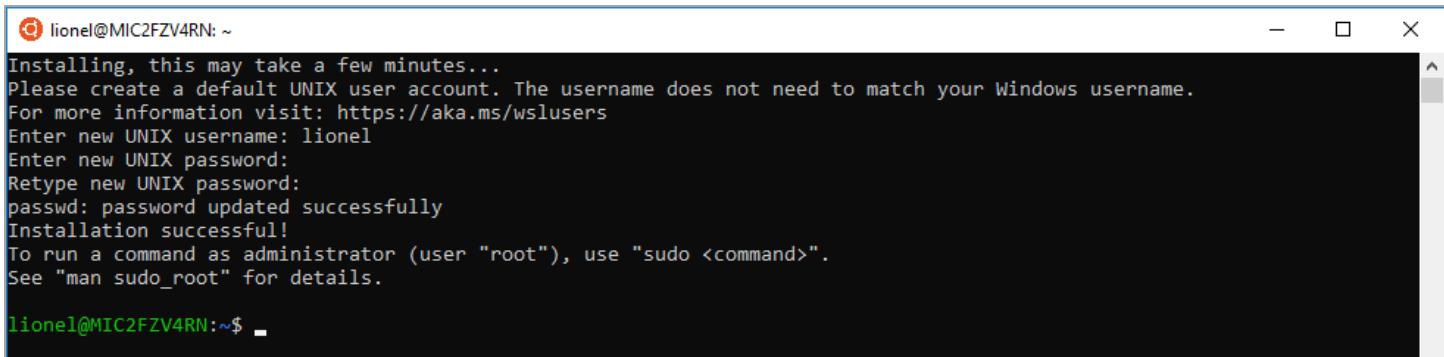
Once installation is complete, you will be prompted to create a new user account (and its password).



This user account is for the normal non-admin user that you'll be logged-in as by default when launching a distro.

**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!



**Note:** Python should be already installed (`python3 --version`)



## Installing Ansible on Ubuntu WSL

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

README.md

build passing coverage 100%

## Ansible Modules for HPE OneView

Modules to manage HPE OneView using Ansible playbooks.

### Requirements

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

- To install Ansible, enter:

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

```
demopaq@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.
```

- Press **Enter** to continue

**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$
```

```
Get:25 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [208 kB]
Get:26 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [21.2 kB]
Get:27 http://security.ubuntu.com/ubuntu bionic-security/restricted Translation-en [5984 B]
Get:28 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [645 kB]
Get:29 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [217 kB]
Get:30 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [6340 B]
Get:31 http://security.ubuntu.com/ubuntu bionic-security/multiverse Translation-en [2640 B]
Fetched 18.4 MB in 26s (709 kB/s)
Reading package lists... Done
demopaq@MIC2FZV4RN:~$
```

- Next, we need to update *apt*, type:

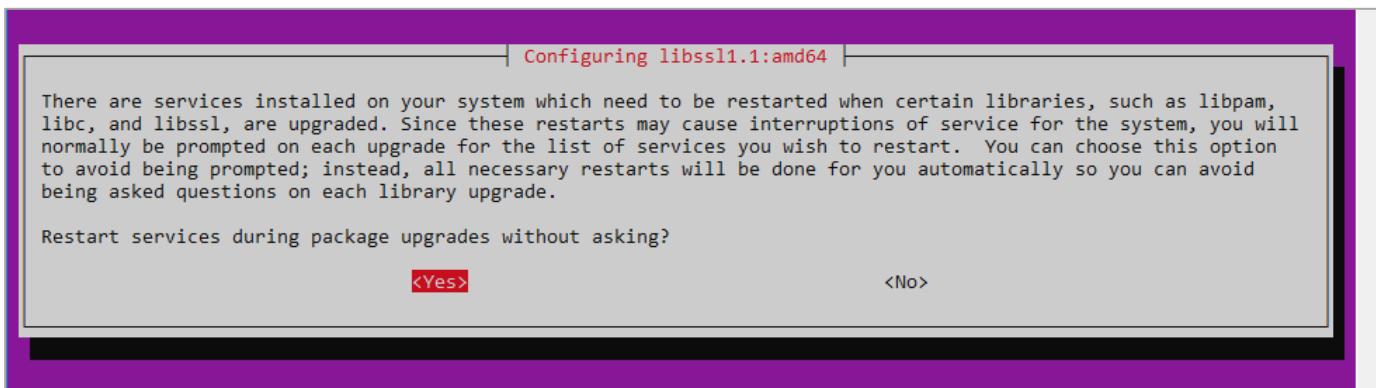
```
sudo apt-get update
```

```
demopaq@MIC2FZV4RN:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
demopaq@MIC2FZV4RN:~$
```

- Then we can install Ansible with:

```
sudo apt-get install ansible -y
```

- If asked, accept any service restart messages:



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

```
ansible --version
```

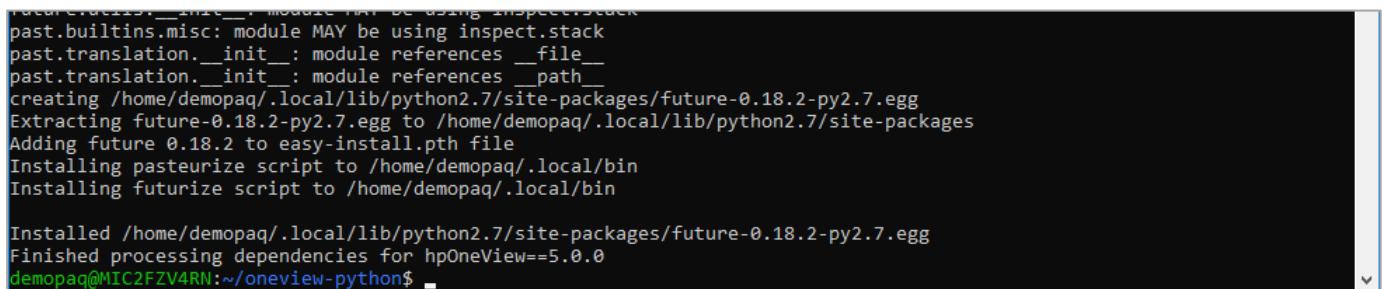
```
demopaq@MIC2FZV4RN:~$ ansible --version
ansible 2.9.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/demopaq/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Nov  7 2019, 10:07:09) [GCC 7.4.0]
demopaq@MIC2FZV4RN:~$
```

## 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:

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



```
past.builtins.misc: module MAY be using inspect.stack
past.translation._init_: module references __file__
past.translation._init_: module references __path__
creating /home/demopaq/.local/lib/python2.7/site-packages/future-0.18.2-py2.7.egg
Extracting future-0.18.2-py2.7.egg to /home/demopaq/.local/lib/python2.7/site-packages
Adding future 0.18.2 to easy-install.pth file
Installing pasteurize script to /home/demopaq/.local/bin
Installing futurize script to /home/demopaq/.local/bin

Installed /home/demopaq/.local/lib/python2.7/site-packages/future-0.18.2-py2.7.egg
Finished processing dependencies for hpOneView==5.0.0
demopaq@MIC2FZV4RN:~/oneview-python$
```

**Important notice:** Do not 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:

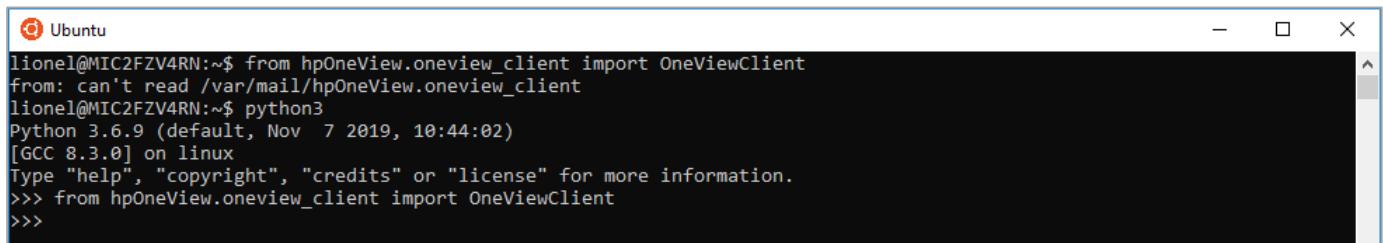
```
git config --global http.proxy http://proxy.myproxy.com:port/
git config --global https.proxy http://proxy.myproxy.com:port/
```

- 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
```



```
Ubuntu
lionel@MIC2FZV4RN:~$ from hpOneView.oneview_client import OneViewClient
from: can't read /var/mail/hpOneView.oneview_client
lionel@MIC2FZV4RN:~$ 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))
```

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

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

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

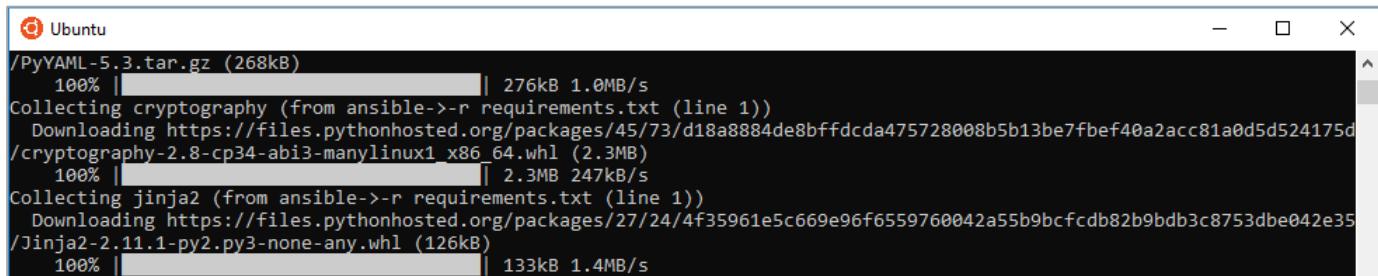
## Installing Ansible Modules for HPE OneView on Ubuntu WSL

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

- Enter:

```
sudo apt-get install python3-pip
cd ..
git clone https://github.com/HewlettPackard/oneview-ansible.git
pip3 install -r requirements.txt
```

The last step will take several minutes, please be patient...



```
/PyYAML-5.3.tar.gz (268kB) 100% | 276kB 1.0MB/s
Collecting cryptography (from ansible->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/45/73/d18a8884de8bffdcd475728008b5b13be7fbef40a2acc81a0d5d524175d/cryptography-2.8-cp34-abi3-manylinux1_x86_64.whl (2.3MB)
    100% | 2.3MB 247kB/s
Collecting jinja2 (from ansible->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/27/24/4f35961e5c669e96f6559760042a55b9bcfcdb82b9bdb3c8753dbe042e35/Jinja2-2.11.1-py2.py3-none-any.whl (126kB)
    100% | 133kB 1.4MB/s
```

**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/
```

```
lionel@MIC2FZV4RN:~/oneview-ansible$ pip3 install -r requirements.txt --proxy http://web-proxy.corp.hpecorp.net:8088
Collecting ansible (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/ec/17/1057cd4fa1c672dde8cc6a2ebc5df29e35170bcd178291c918b435f/ansible-2.9.5.tar.gz (14.2MB)
    0% | 81kB 151kB/s eta 0:01:34
```

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 new 5.0.0 release. As we need this fix for one of our demos, we need to merge the branch with the fix to our cloned directory.

- At the moment, we have only one branch Master:

```
git branch
```

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

- Enter the following to navigate to the fix branch:

```
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.

```
git branch
```

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

**Note:** The asterisk shows the currently active branch.

- Then we run

```
git checkout master
```

to change the active branch back to *master*. Then we run the command

```
git merge bug_fix/create_profiles_in_parallel
```

to merge the *bug\_fix* features into the *master* branch.

**Note:** *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"
```



# Hewlett Packard Enterprise

- The following nano editor should open:

```
Ubuntu /home/lionel/oneview-ansible/.git/MERGE_MSG
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.
```

[ Read 7 lines ]

^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-E Redo M-G 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$
```



# Hewlett Packard Enterprise

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's modify the Ubuntu user profile.

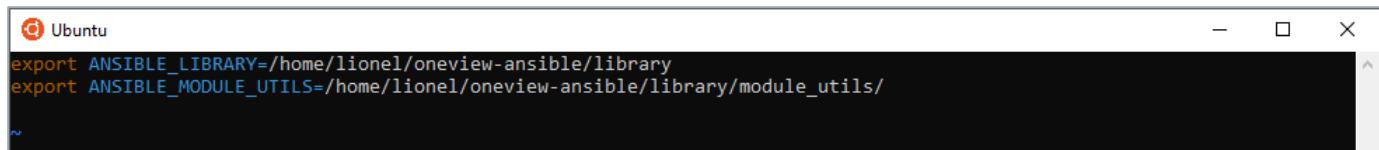
- Type:

```
vi ~/.bash_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.



```
Ubuntu
source ~/.bash_profile
```

- To immediately apply the new environment variables, type:

```
source ~/.bash_profile
```

- 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": 1200,
    "credentials": {
        "userName": "administrator",
        "authLoginDomain": "",
        "password": "password"
    }
}
EOF
```



## Installing Git for VS Code source control

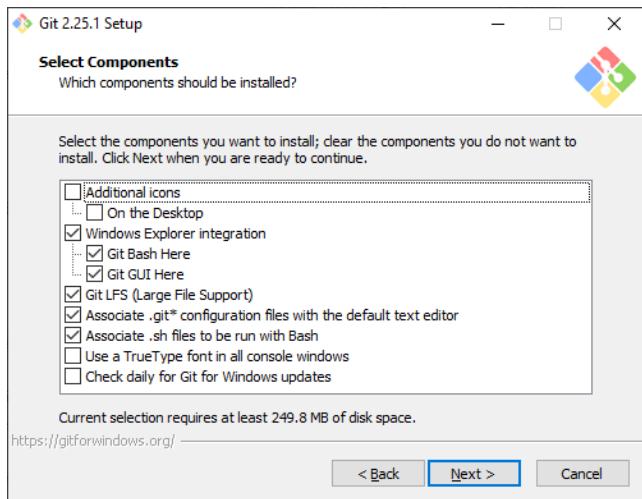
Visual Studio Code does come with an integrated Git source control provider. However, for that to work, Git itself needs to be installed on your Windows system as well.

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:



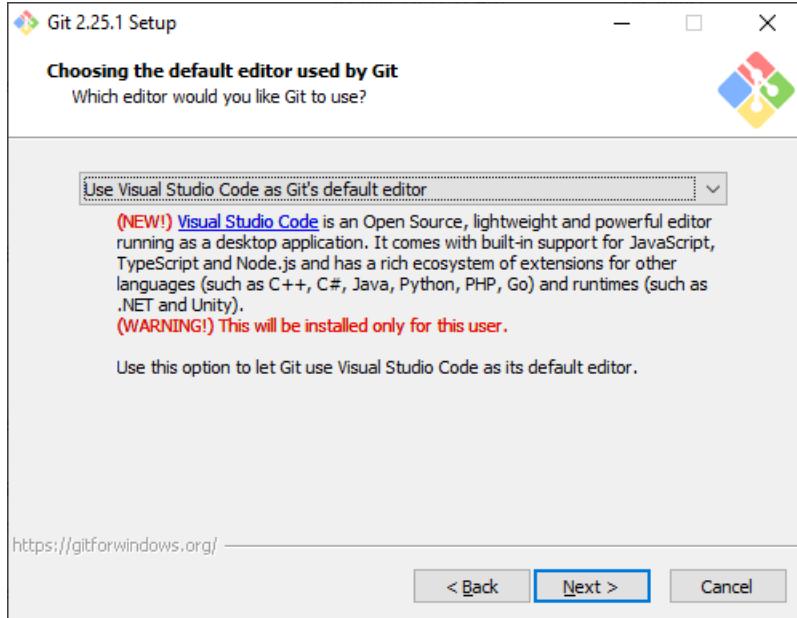
- Keep the proposed default components selected



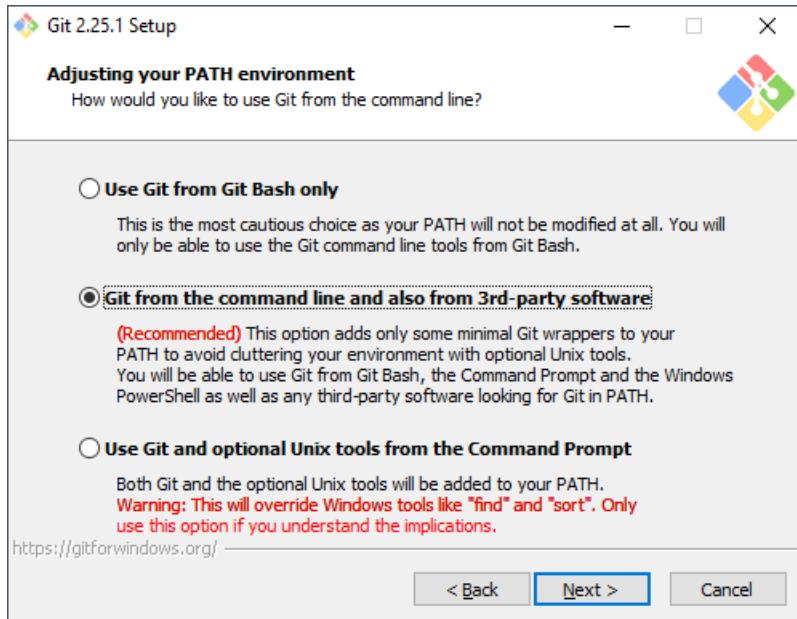


# Hewlett Packard Enterprise

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

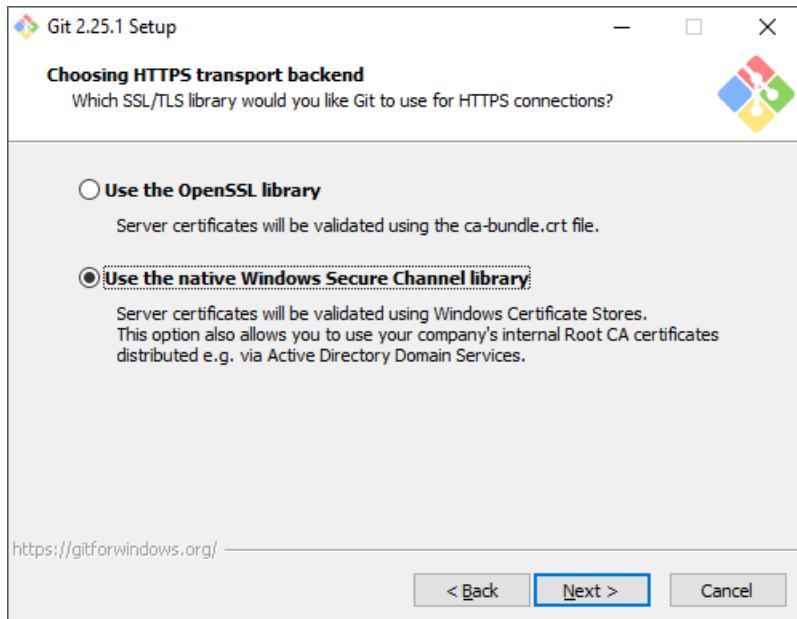


- Keep the recommended PATH environment option selected

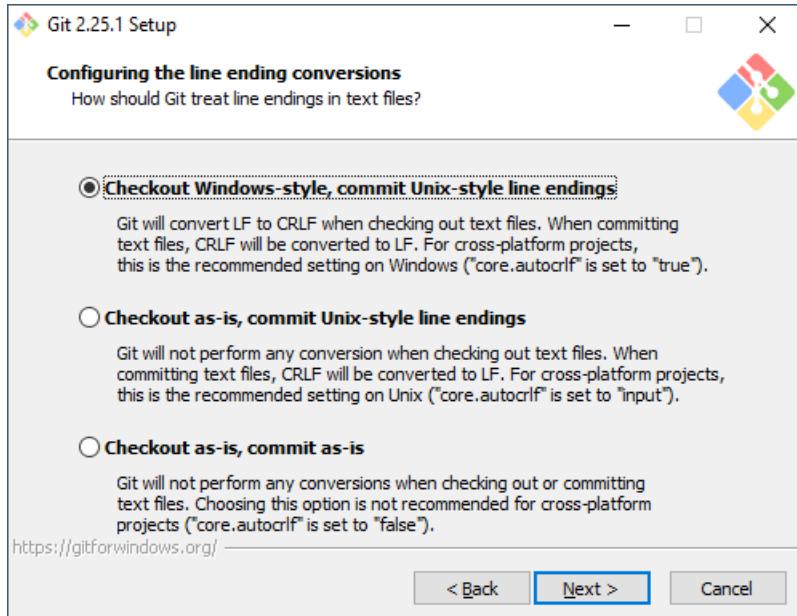


# Hewlett Packard Enterprise

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



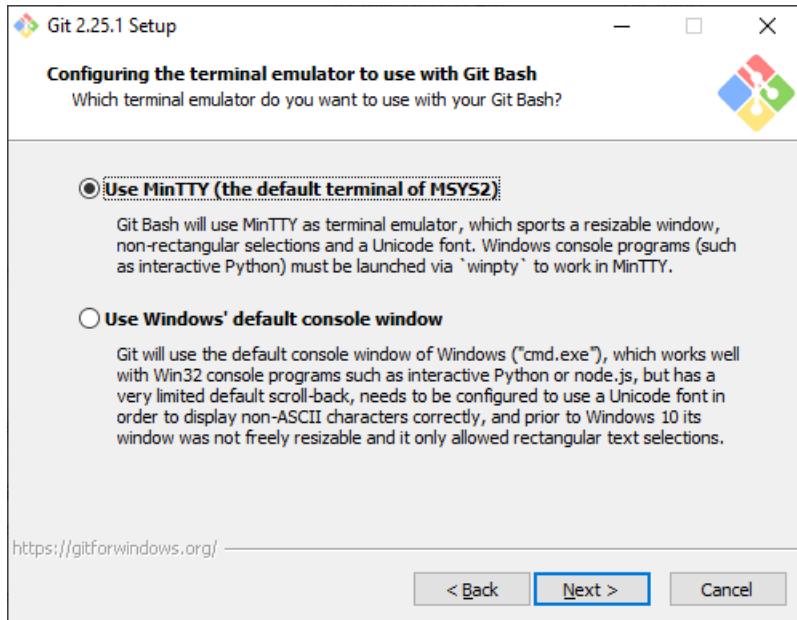
- Use the **Checkout Windows-style** as it is the recommended setting on Windows:



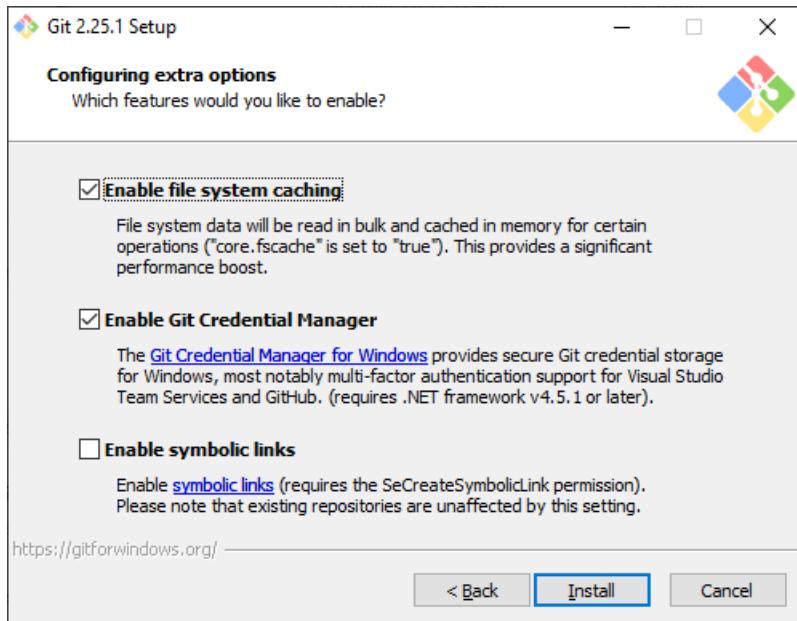


# Hewlett Packard Enterprise

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



- Keep the default extra options



Once the installation is completed, 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.

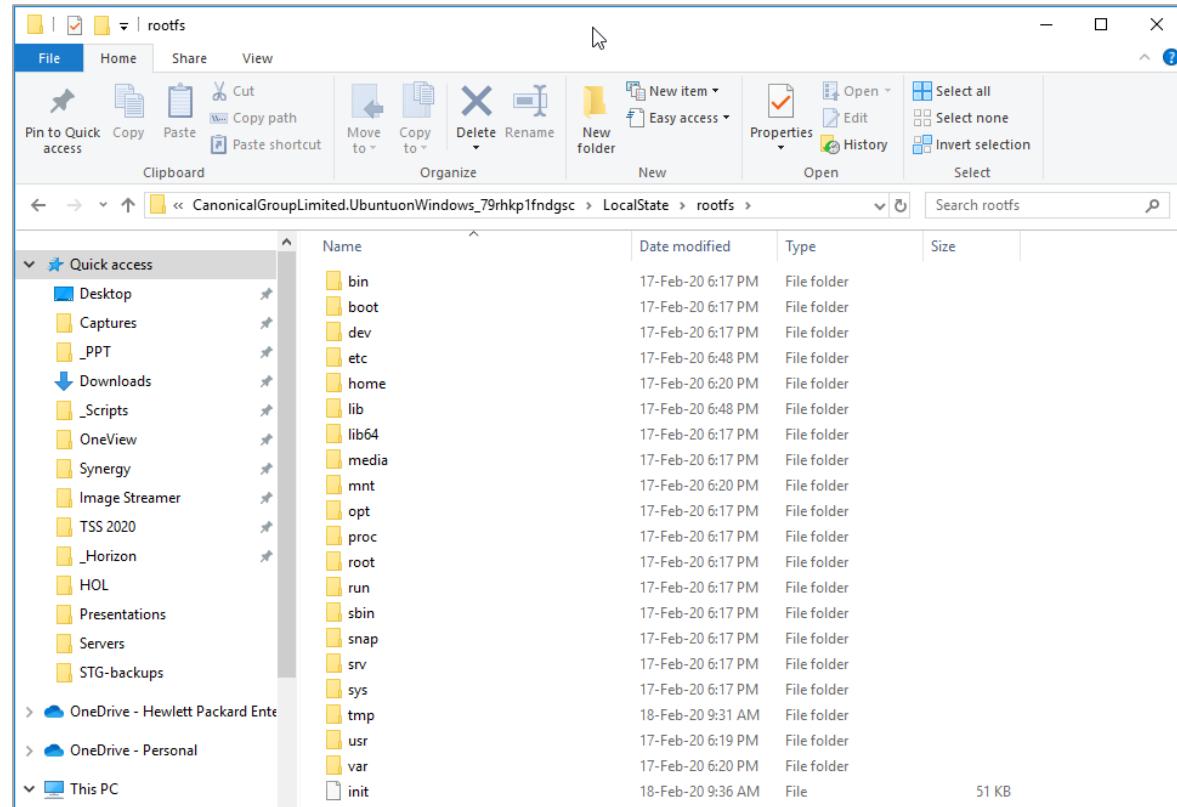
/mnt/c/ shortcut maps to c:\ Windows drive

```
lionel@MIC2FZV4RN:~/oneview-ansible$ cd /mnt/c/
$GetCurrent/                               PerfLogs/
$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\





# Hewlett Packard Enterprise

**Warning:** Don't open/edit any files here from Windows. Just ignore the files here. Editing any files from Windows can damage your Linux distro.

**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.

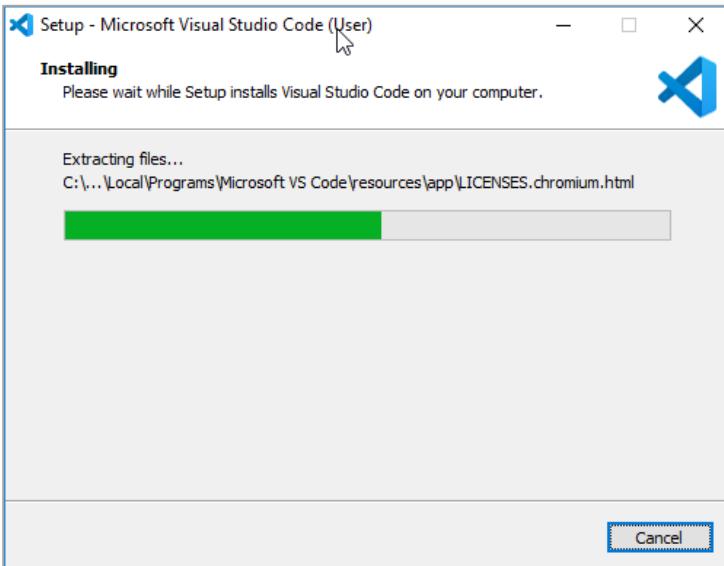
The screenshot shows the official Visual Studio Code download page. At the top, there's a large "Download Visual Studio Code" button. Below it, a sub-header says "Free and built on open source. Integrated Git, debugging and extensions." There are three main download sections: one for Windows (Windows 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 blue "Windows", ".deb", ".rpm", or "Mac" button with a downward arrow. Under each button, there are two options: "User Installer" and "System Installer" (Windows and Mac) or ".zip" and ".tar.gz" (Linux). Below the ".zip" and ".tar.gz" links is a "Snap Store" link. The Windows icon is the blue Microsoft logo, the Linux icon is Tux the Penguin, and the Mac icon is the Apple logo.

Platform	Installer Type	Architectures
Windows	User Installer	64 bit 32 bit
	System Installer	64 bit 32 bit
	.zip	64 bit 32 bit
Linux	.deb	64 bit
	.rpm	64 bit
	.tar.gz	64 bit
Mac	Snap Store	



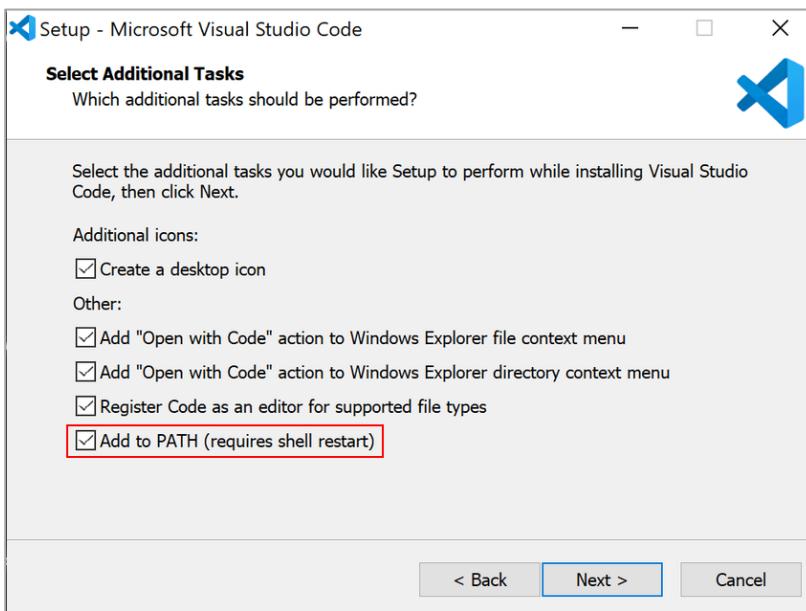
# Hewlett Packard Enterprise

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



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

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



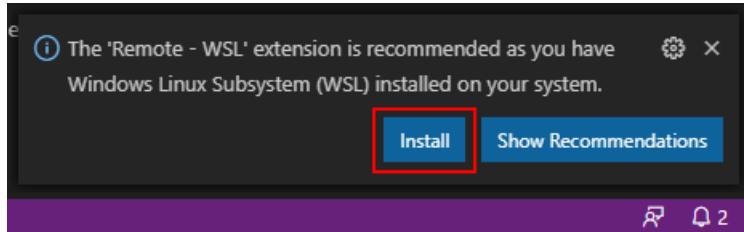


# Hewlett Packard Enterprise

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

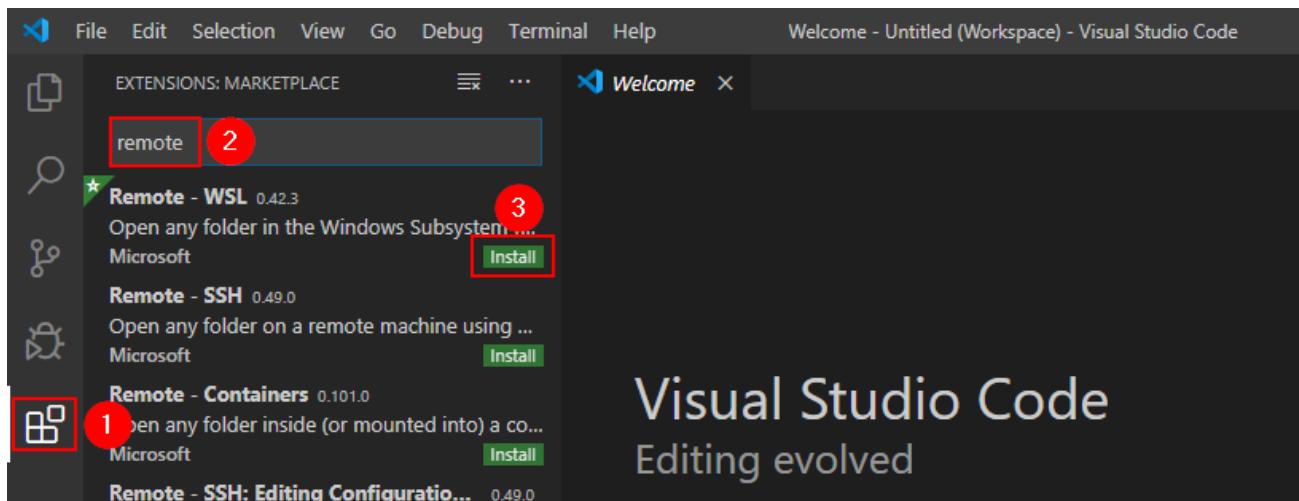
When VS Code is opening, you should notice a warning message saying "The 'Remote - WSL' extension is recommended as you have Windows Linux Subsystem (WSL) installed on your system"

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



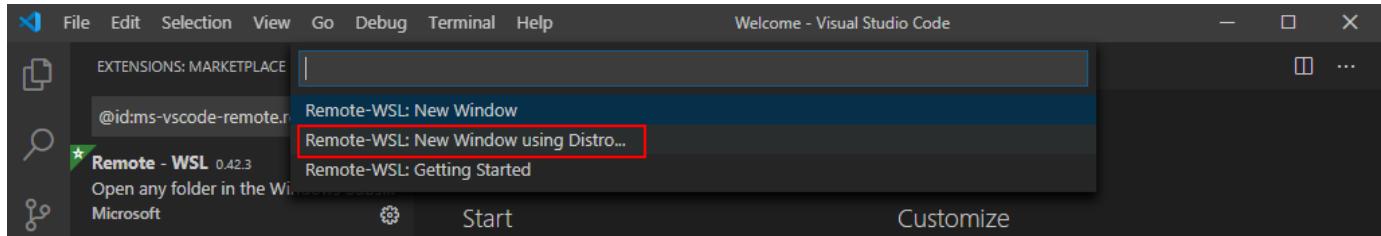


**Hewlett Packard  
Enterprise**

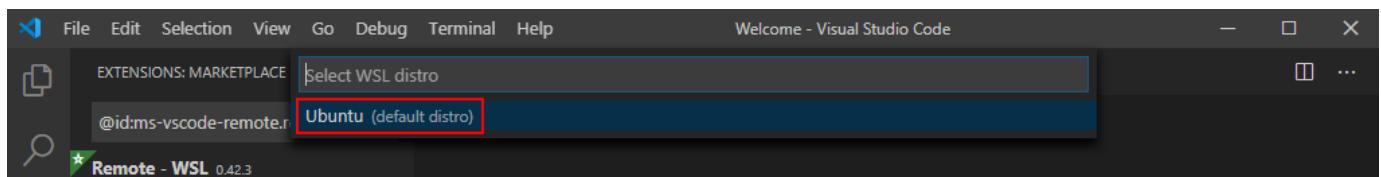


# Hewlett Packard Enterprise

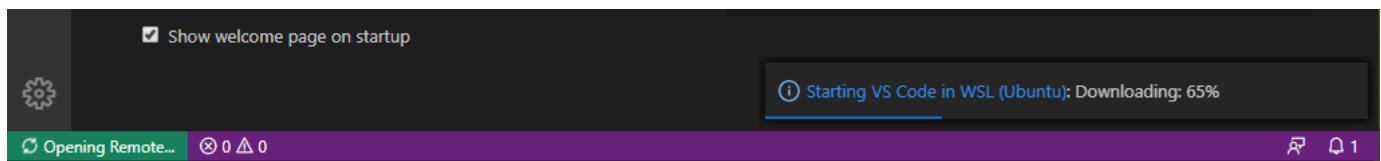
- Select **Remote-WSL: New Window using Distro...**



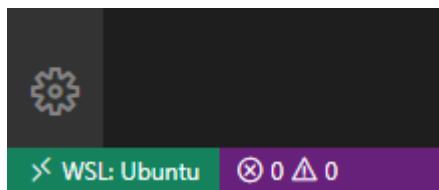
- Then select **Ubuntu**



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



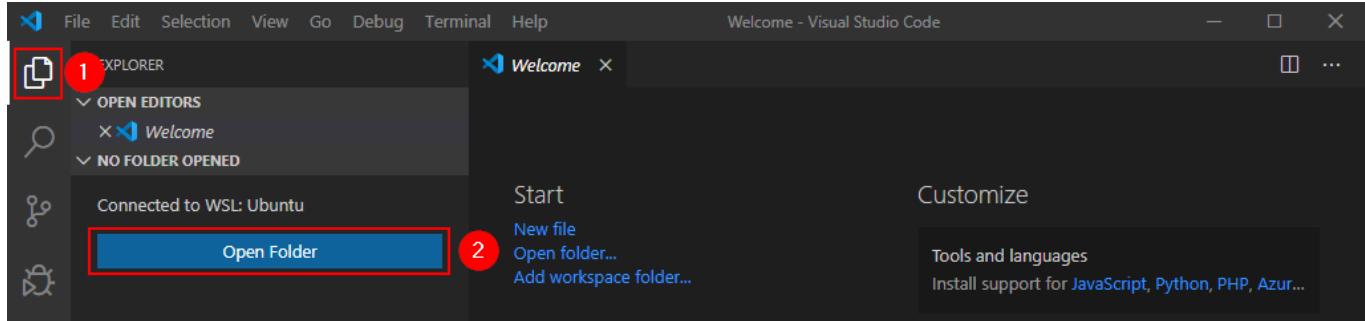
- You can close the previous VS Code window.
- Once connected, in the lower left corner of the Status Bar, you should see that you're connected to your WSL: Ubuntu instance.



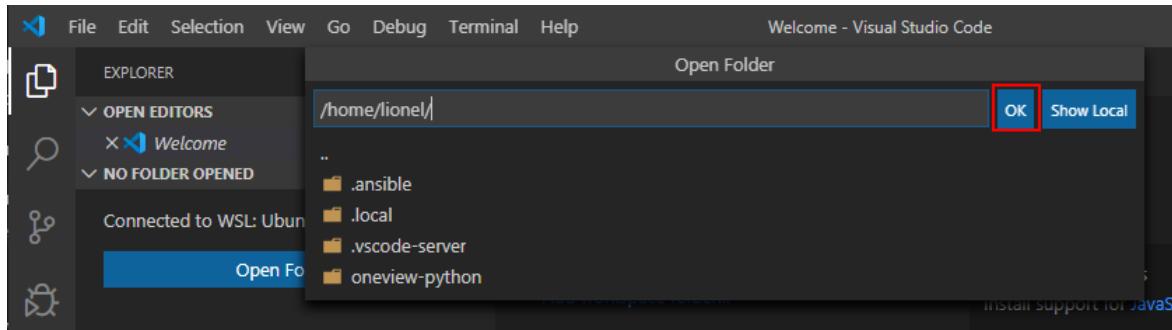


# Hewlett Packard Enterprise

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



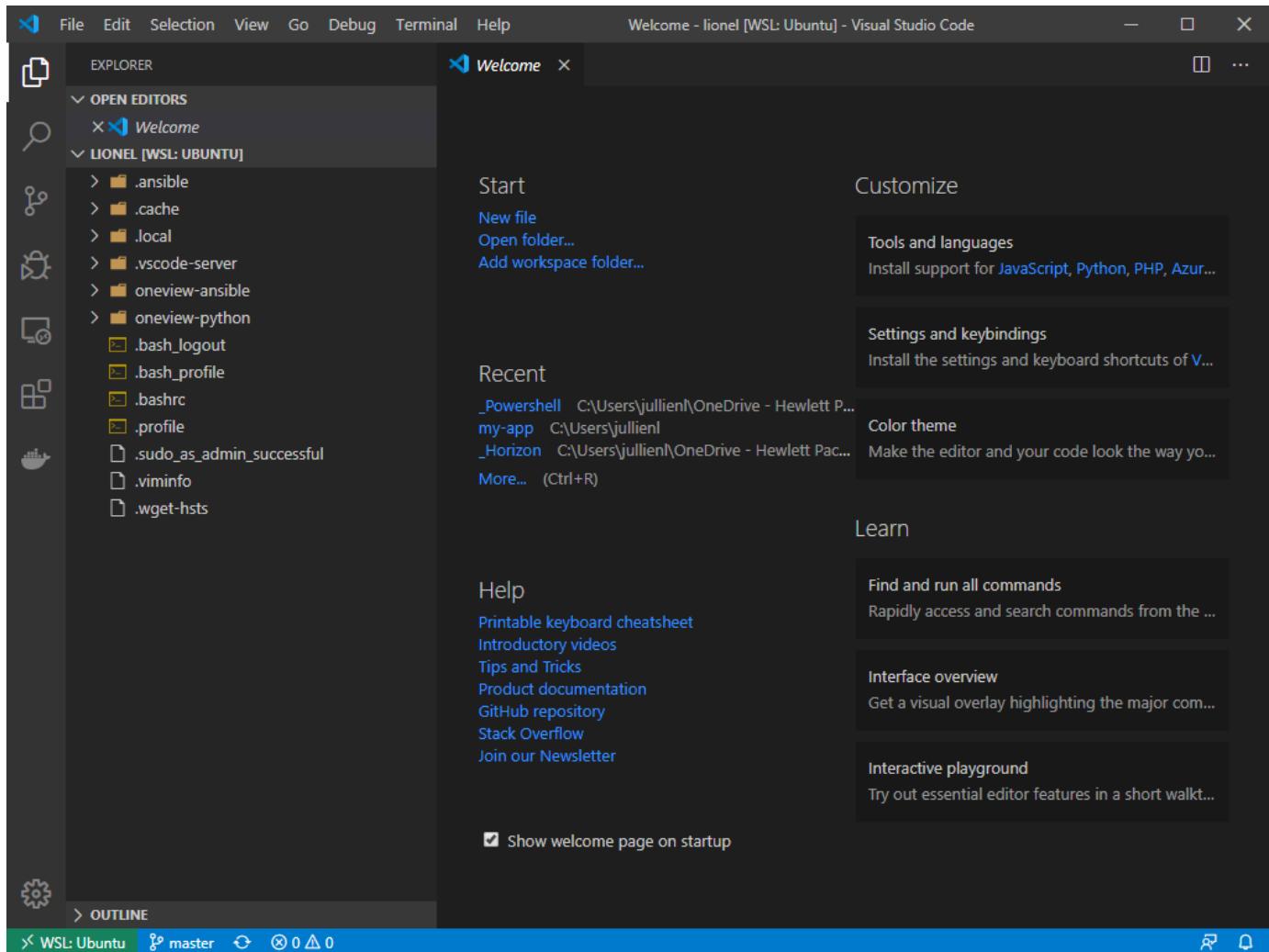
- Click **OK** to the /home/<username> folder





# Hewlett Packard Enterprise

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





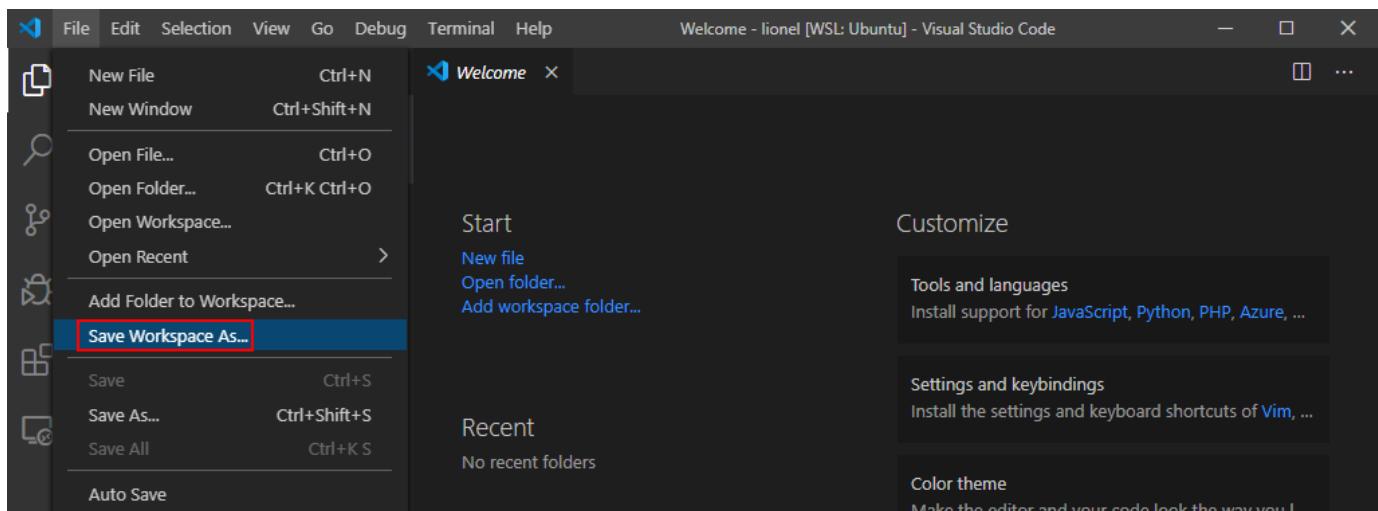
# Hewlett Packard Enterprise

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 screenshot shows the VS Code settings search interface. The search bar at the top contains the text "formaton". Below the search bar, there are tabs for "User", "Workspace", "PowerShell", and "Folder". The results are grouped under "Commonly Used (1)". The first result is "Text Editor (6)" which includes "Cursor (2)", "Formatting (3)", and "Features (2)" which includes "Terminal (2)". The second result is "Extensions (46)" which includes "CSS (9)", "Emmet (1)", "HTML (10)", "JSON (1)", "LESS (6)", "PowerShell Conf... (2)", "SCSS (Sass) (7)", and "TypeScript (10)". The third result is "JavaScript > Format: Place Open Brace On New Line For Control Blocks". The fourth result is "JavaScript > Format: Place Open Brace On New Line For Functions". The "Editor: Format On Save" setting is highlighted with a red box. It is located under "Features (2)" and is described as "Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down." The status bar at the bottom right of the interface shows "55 Settings Found".

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.

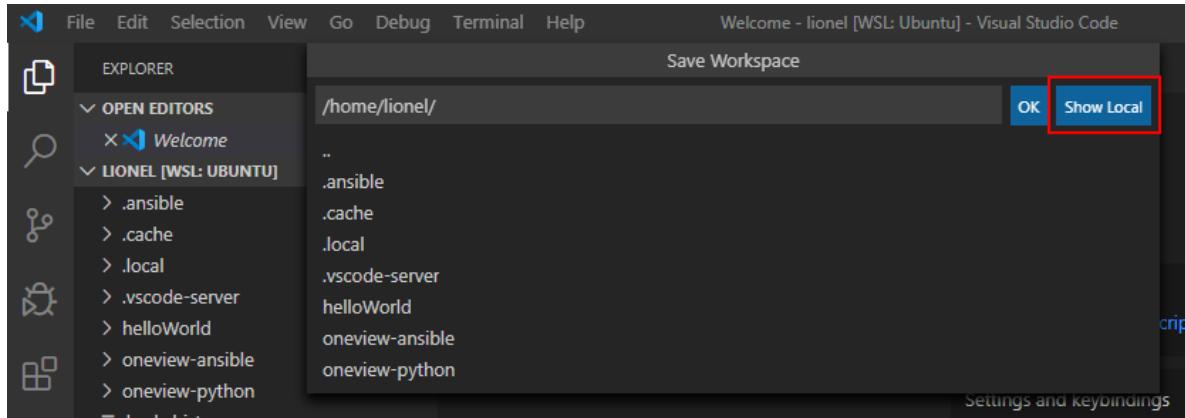
- Select **File** menu and click on **Save Workspace as...**:



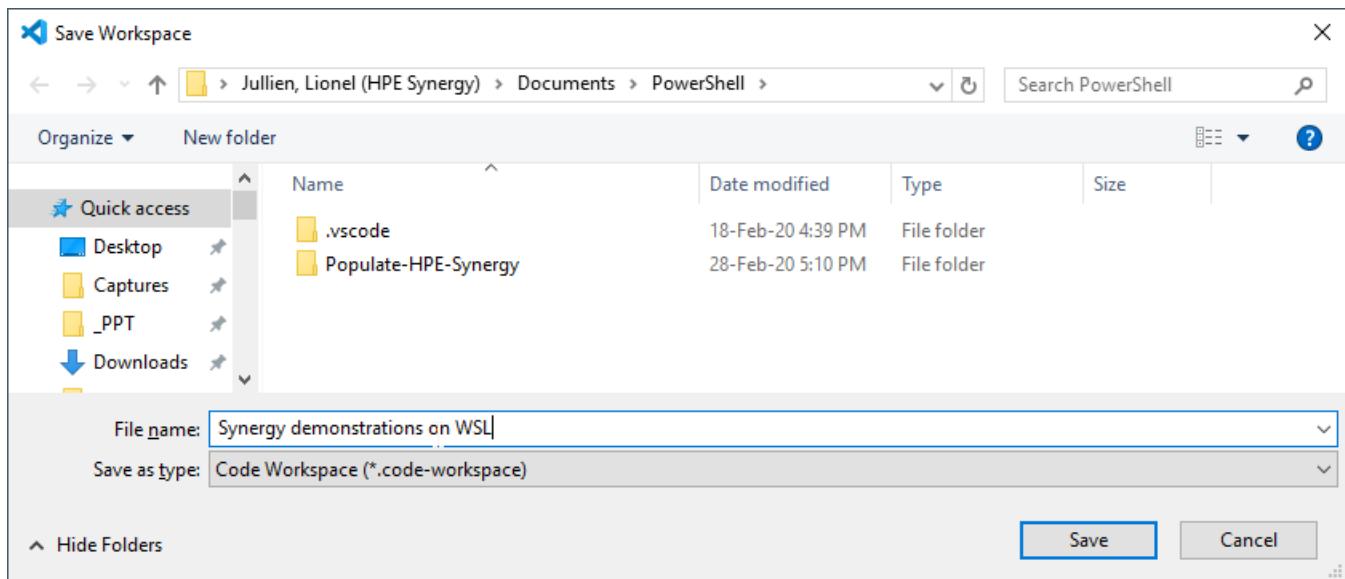


# Hewlett Packard Enterprise

- 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.

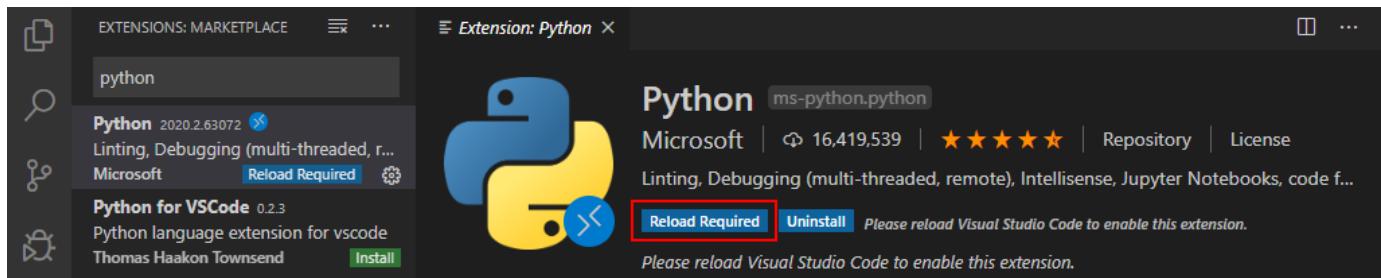
There are popular ones for PowerShell, Python, Ansible...

- 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 completed, click on **Reload Required** to activate the extension.

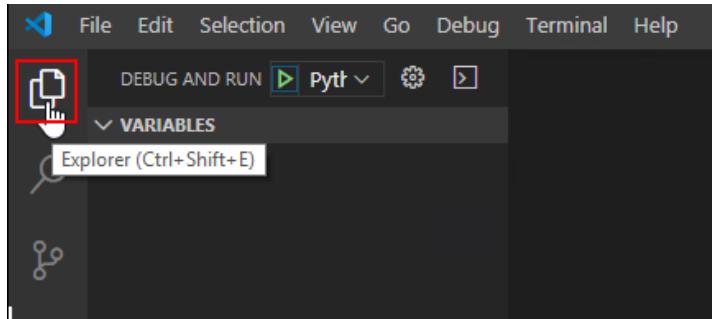


**Note:** For the Python extension, we are going to use the Python interpreter running in our Ubuntu Linux distribution hosted by WSL (Windows Subsystem for Linux). Another good option if you don't want to use WSL would be to run Python directly on your Windows system, see <https://www.python.org/downloads/windows/>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

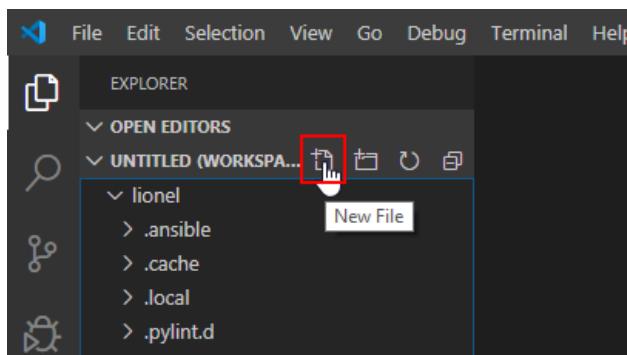


# Hewlett Packard Enterprise

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

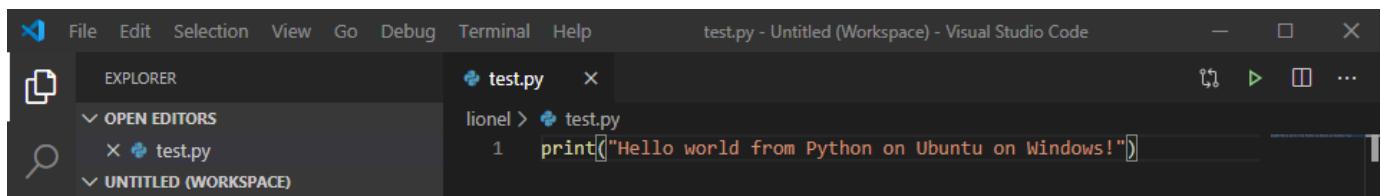


- Then click on **New File**



- Enter a name like **test.py** then press **ENTER**
- Enter the following line to build the Python script test.

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

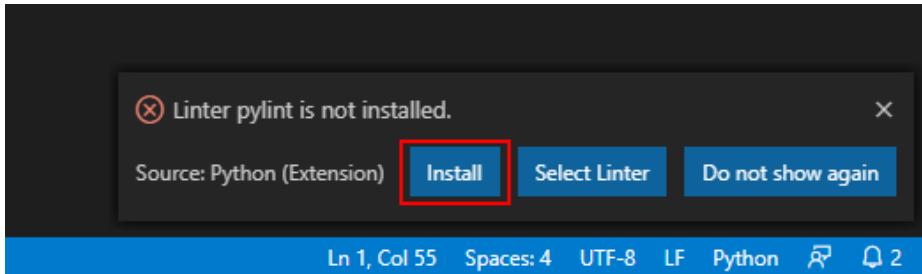


- Save the file by pressing **CTRL + S**



# Hewlett Packard Enterprise

- You can install pylint by clicking on **Install** when the windows pops-up at the creation of the Python script:

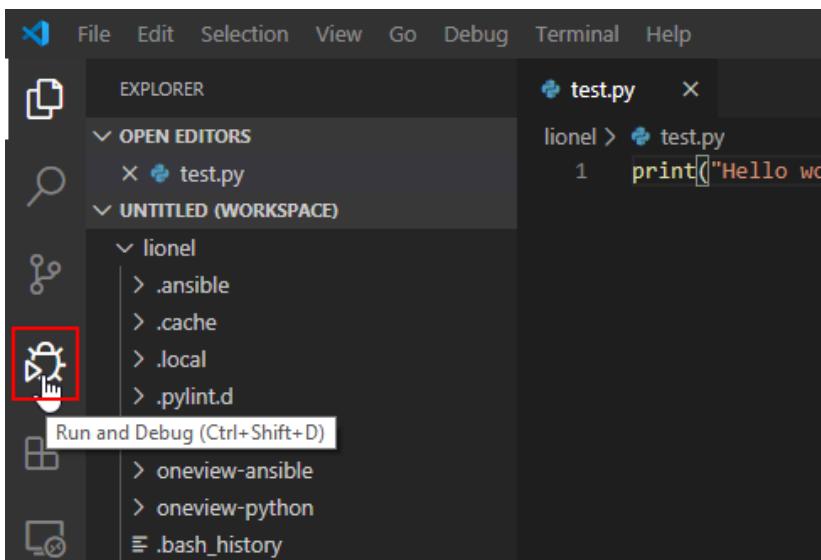


Pylint is a Python static code analysis tool which looks for programming errors, helps enforcing a coding standard, sniffs for code smells and offers simple refactoring suggestions.

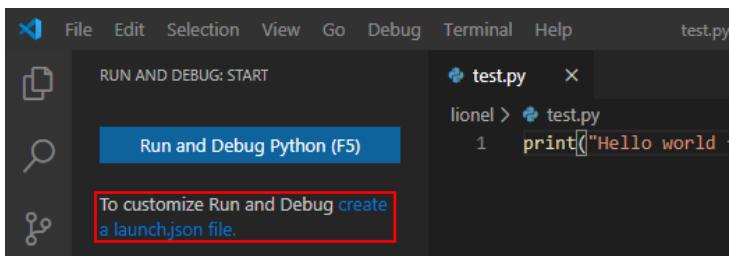
**Note:** Behind a proxy, make sure you add `--proxy` to the command:

```
/usr/bin/python3 -m pip install -U pylint --user --proxy http://web-proxy.corp.hpecorp.net:8088
```

- Click on **Run and Debug**



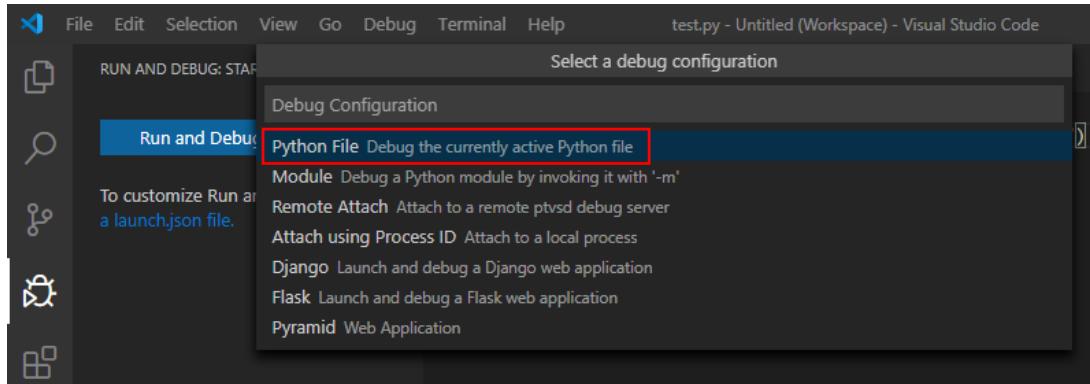
- Select **Customize Run and Debug create a launch.json file**



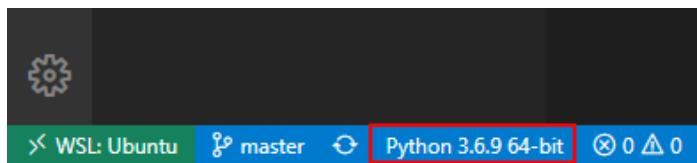


# Hewlett Packard Enterprise

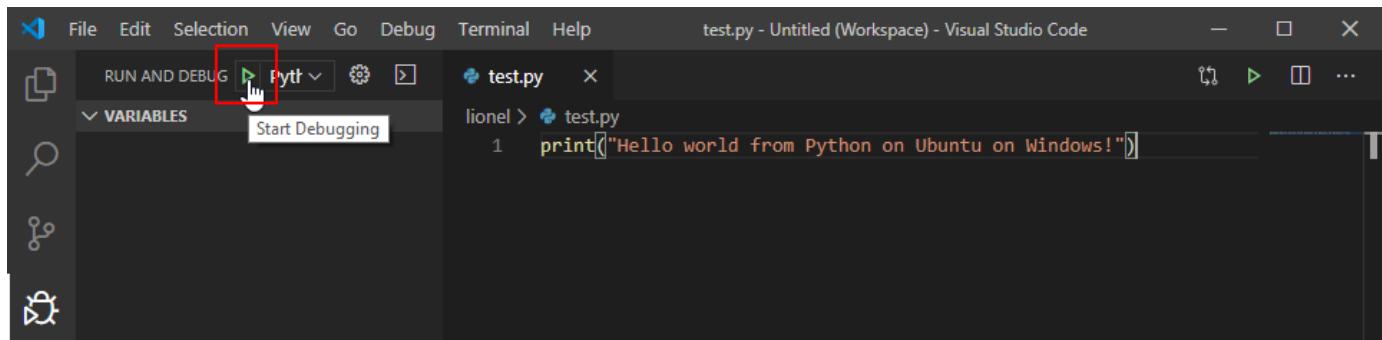
- Select **Python File**



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



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





# Hewlett Packard Enterprise

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

A screenshot of the Visual Studio Code interface. The top bar shows "File Edit Selection View Go Debug Terminal Help test.py - Synergy demonstrations on WSL (Workspace) - Visual Studio Code". The left sidebar has icons for RUN AND DEBUG, VARIABLES, WATCH, CALL STACK, and BREAKPOINTS. The main editor area shows a Python file named "test.py" with the code: 

```
lionel > test.py
1   print("Hello world from Python on Ubuntu on Windows!")
```

 The bottom right terminal window shows the output of running the script: 

```
lionel@MIC2FZV4RN:~$ /usr/bin/python3 /home/lionel/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lionel/test.py
Hello world from Python on Ubuntu on Windows!
lionel@MIC2FZV4RN:~$
```

 The terminal tab is labeled "3: Python Debug Consc". The status bar at the bottom indicates "WSL: Ubuntu master 0 2t Python 3.6.9 64-bit 0 0 Python: Current File (lionel) Ln 1, Col 55 Spaces: 4 UTF-8 LF Python ⚡ 1".

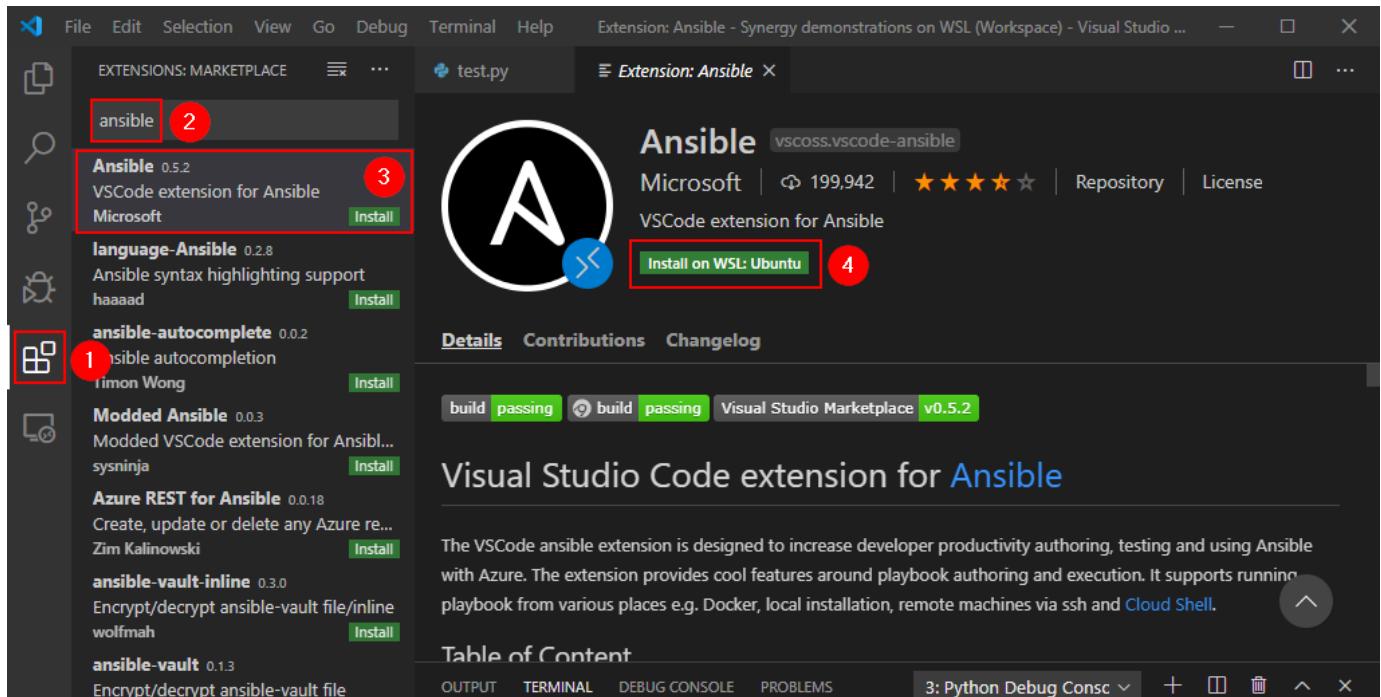
We are now all set for Python scripting and debugging.



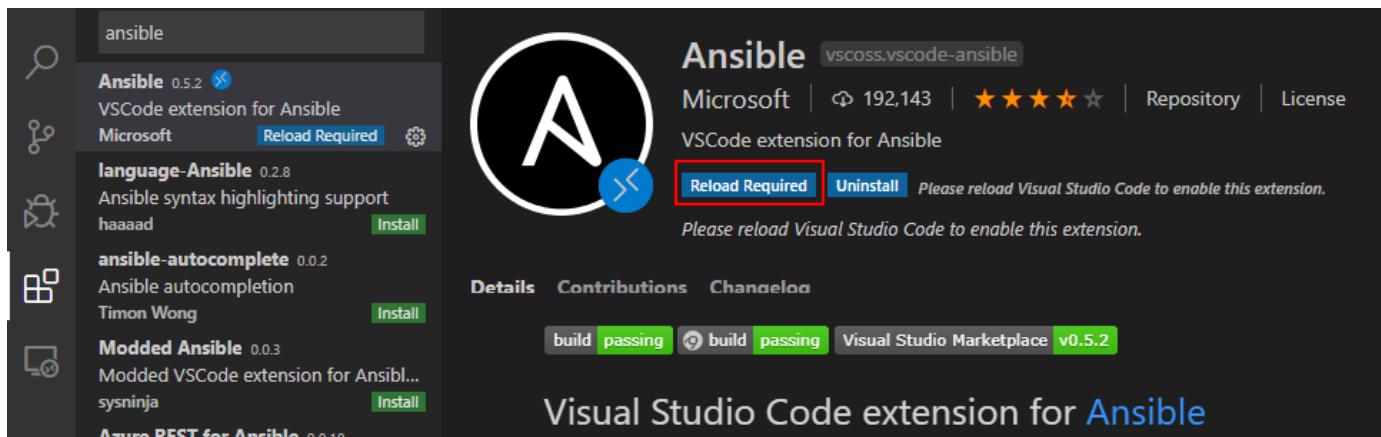
## Preparing VS Code for Ansible

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 completed, click on **Reload Required** to activate the extension.





## Preparing VS Code for PowerShell

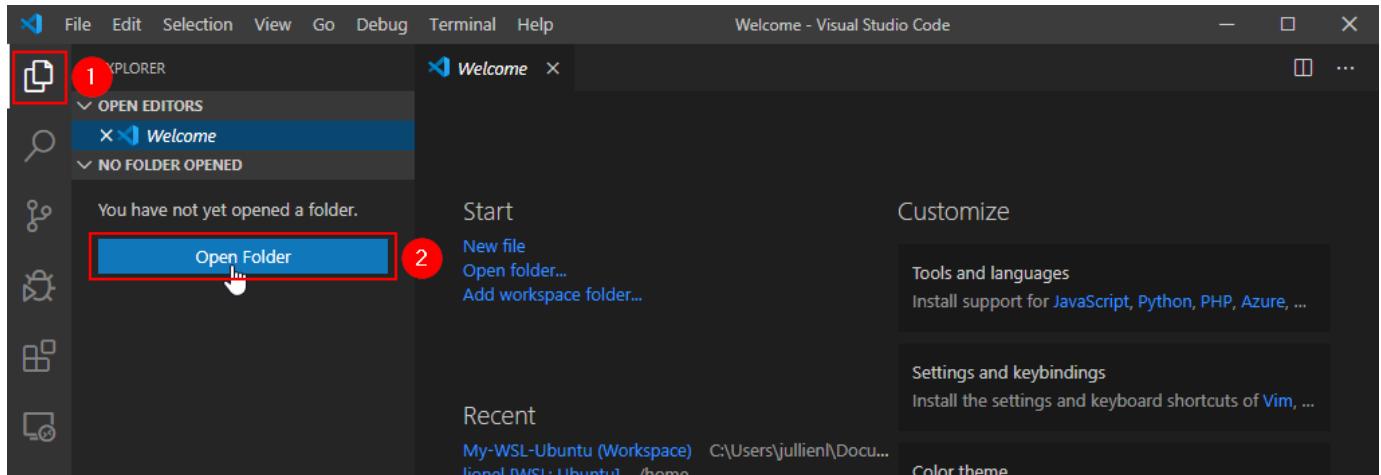
Next, we can install the PowerShell extension to get also nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to cmdlet documentation, ability to run the active script file in the VS Code terminal console, etc.

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**.

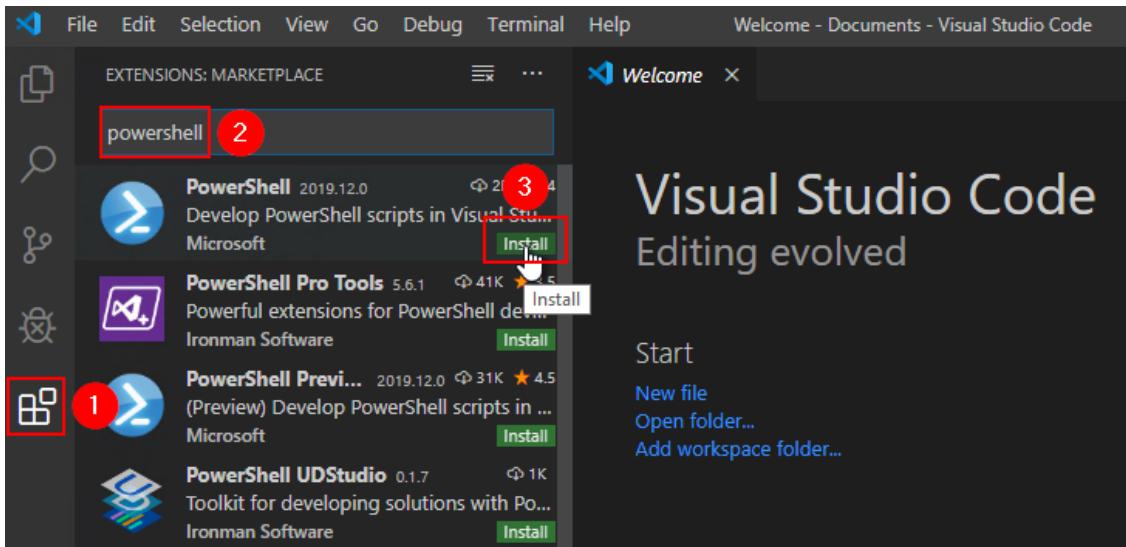


- Open the folder you want your PowerShell project to be in (e.g. \Documents\PowerShell) and click **Select Folder**.

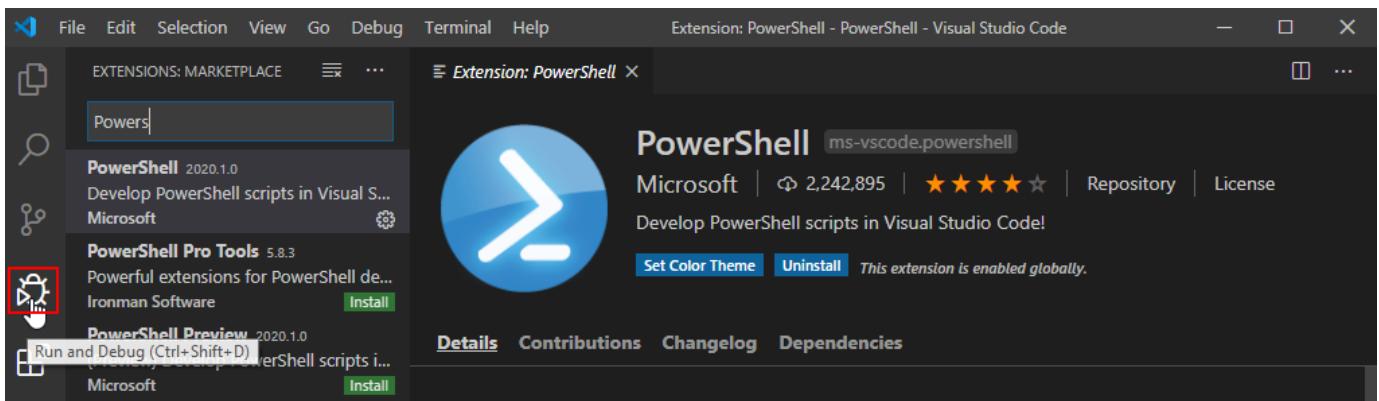


# Hewlett Packard Enterprise

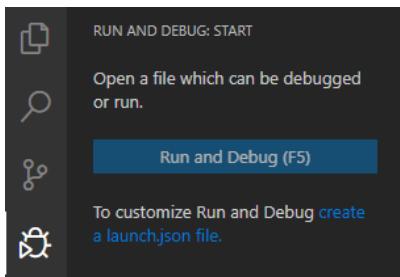
- Go to the **Extensions** pane and search for the **PowerShell** extension and click on the **Install** button:



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



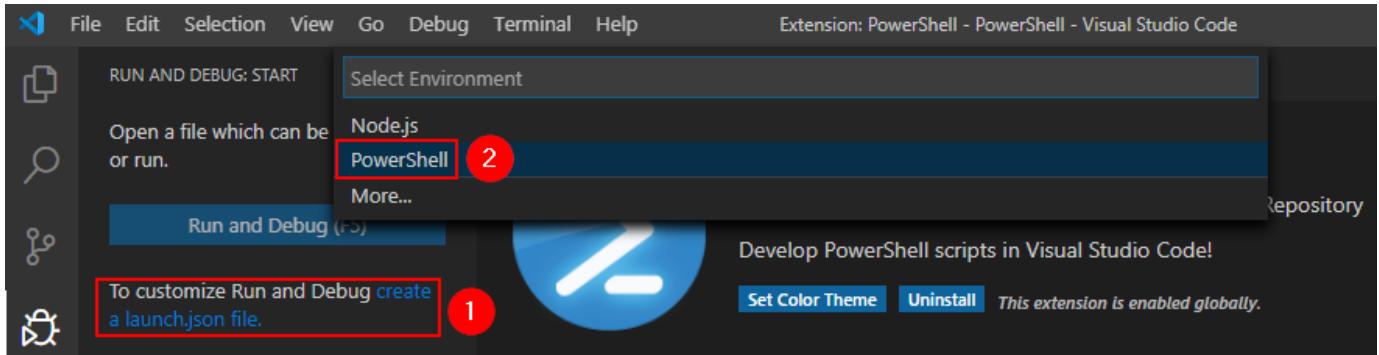
Visual Studio Code's built-in debugger needs some quick setup to run and debug PowerShell scripts. If your system is already configured, you can skip this section. When VS Code is not configured for PowerShell, the following dialog is displayed:



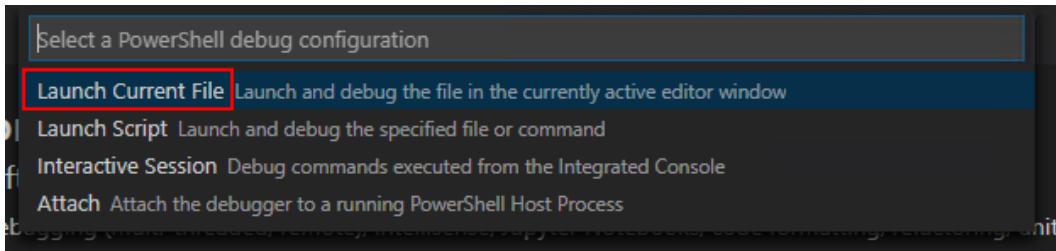


# Hewlett Packard Enterprise

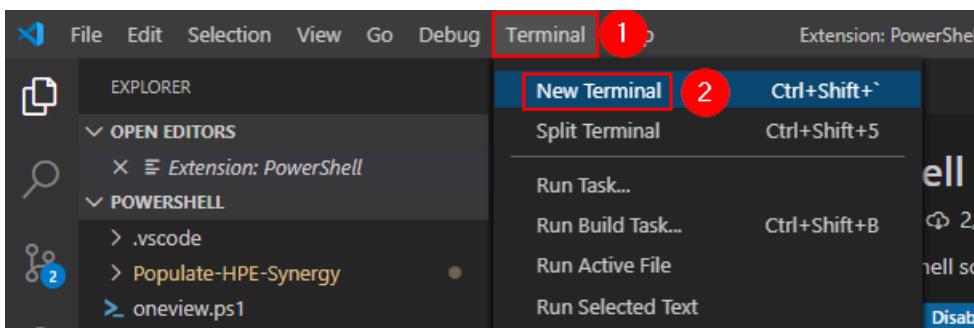
- Click on **Create a launch.json file** to configure Run and Debug, then select **PowerShell**



- 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**



# Hewlett Packard Enterprise

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

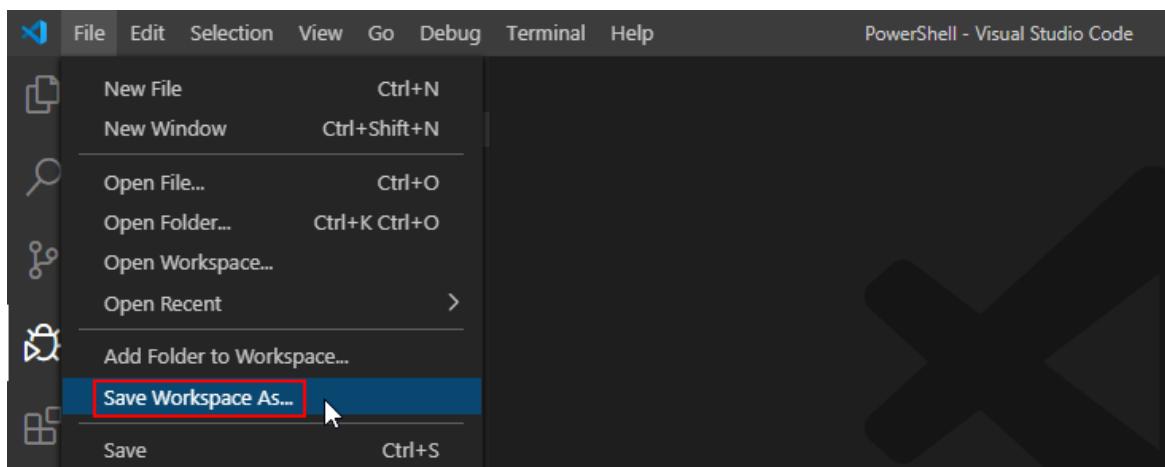
```
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 2: PowerShell Integrated Terminal + □ ✎ ^ ×

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

PS C:\Users\jullien1\Documents\PowerShell> Get-Service
```

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

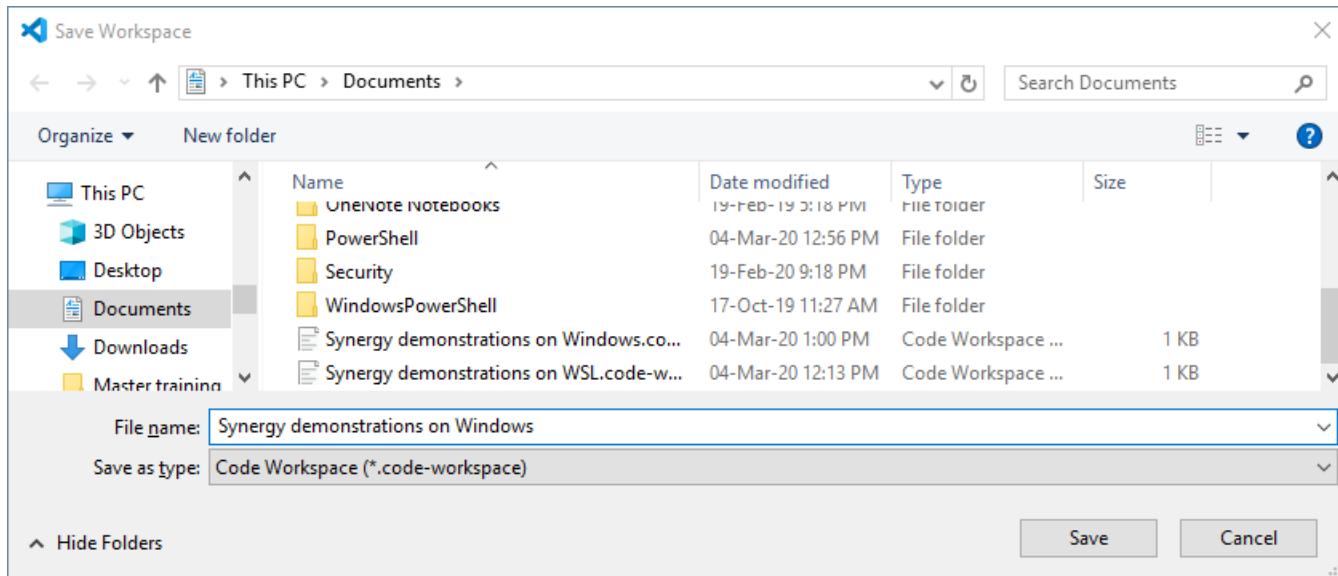
- Select the **File** menu then select **Save Workspace As...**



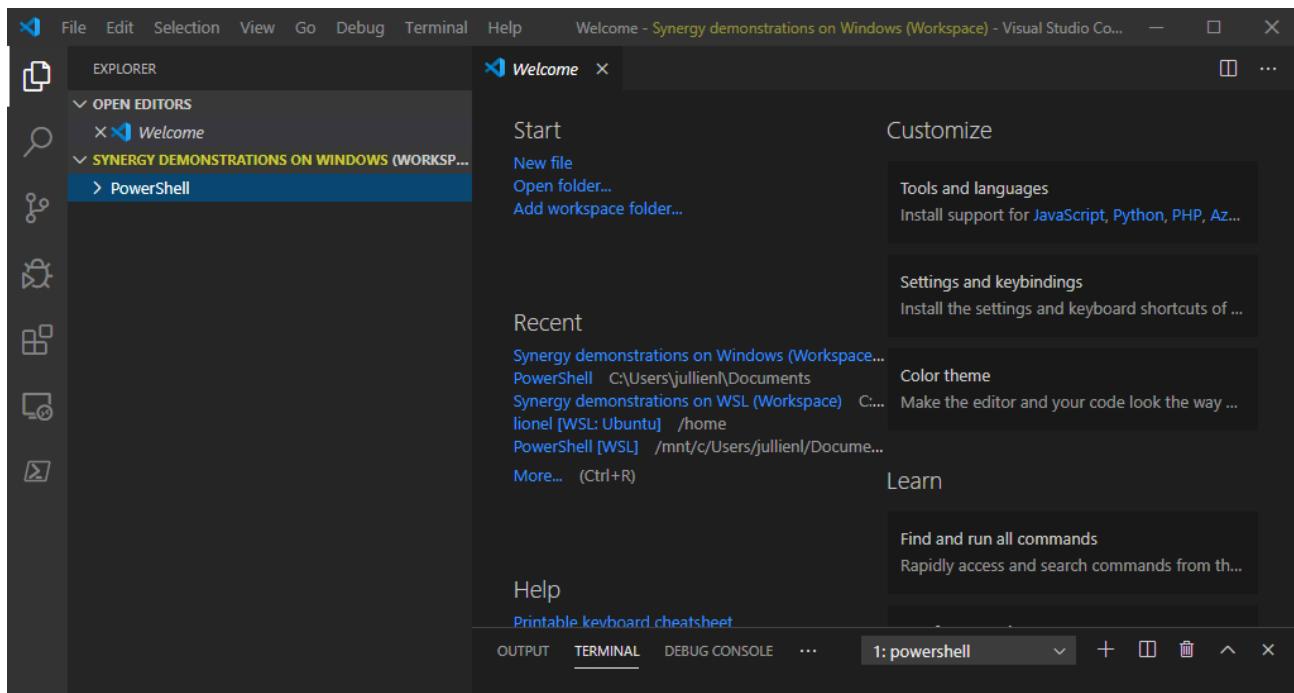


# Hewlett Packard Enterprise

- 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.



# Hewlett Packard Enterprise

## 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**

The screenshot shows the GitHub repository page for 'HewlettPackard / POSH-HPOneView'. At the top, there are navigation links for 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. On the right, there are buttons for 'Search', 'Sign in', and 'Sign up'. Below the header, the repository name 'HewlettPackard / POSH-HPOneView' is displayed, along with statistics: 68 watches, 86 stars, and 36 forks. A 'Watch' button, a 'Star' button, and a 'Fork' button are shown. The 'Code' tab is selected. Below the tabs, there are links for 'Issues 45', 'Pull requests 0', 'Projects 1', 'Wiki', 'Security', and 'Insights'. The main content area describes the repository as a 'PowerShell language bindings library for HPE OneView.' It includes a link to the documentation: <http://hewlett-packard.github.io/POSH-...>. Below this, there is a list of tags: 'powershell', 'hpe-oneview', 'infrastructure', 'infrastructure-as-code', 'composable-infrastructure', 'oneview', and 'hpe-synergy'. Further down, there are metrics: 154 commits, 11 branches, 0 packages, and 111 releases (which is highlighted with a red box). It also shows 1 contributor and the MIT license. At the bottom, there are buttons for 'Branch: 5.00', 'New pull request', 'Find file', and 'Clone or download'. A commit history section shows a recent commit by 'ChrisLynchHPE' fixing a README, dated 'Nov 6, 2019'.

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.500
```

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The title bar indicates '2: PowerShell Integrated'. The terminal window displays the command 'Install-Module HPOneView.500' being run. The output shows several messages: 'NuGet provider is required to continue', 'PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or 'C:\Users\Administrator.1j\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PoweYeSelliGet to install and import the NuGet provider now?', 'Untrusted repository: [?] Help (default is "Yes"):', '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"):', and finally 'PS C:\Users\Administrator.1j\Documents\DCS Appliance>'. The terminal has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'.



# Hewlett Packard Enterprise

- 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.500
```

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

```
get-hpovversion
```

```
PS C:\Users\jullienl\Documents\PowerShell> Get-HPOVersion
LibraryVersion Path
-----
5.0.2295.3359 C:\Users\jullienl\Documents\WindowsPowerShell\Modules\hponeview.500\5.0.2295.3359
```

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. Enter:

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

or

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

#### Note:

If an old version of the module is already installed on your system, use the following commands to update to the latest version:

```
Get-Module HPOneView.500 -ListAvailable | Uninstall-Module
Install-Module -Name HPOneView.500
```

This completes the PowerShell configuration and concludes Chapter-3

In the next chapter, we will configure HPE OneView.

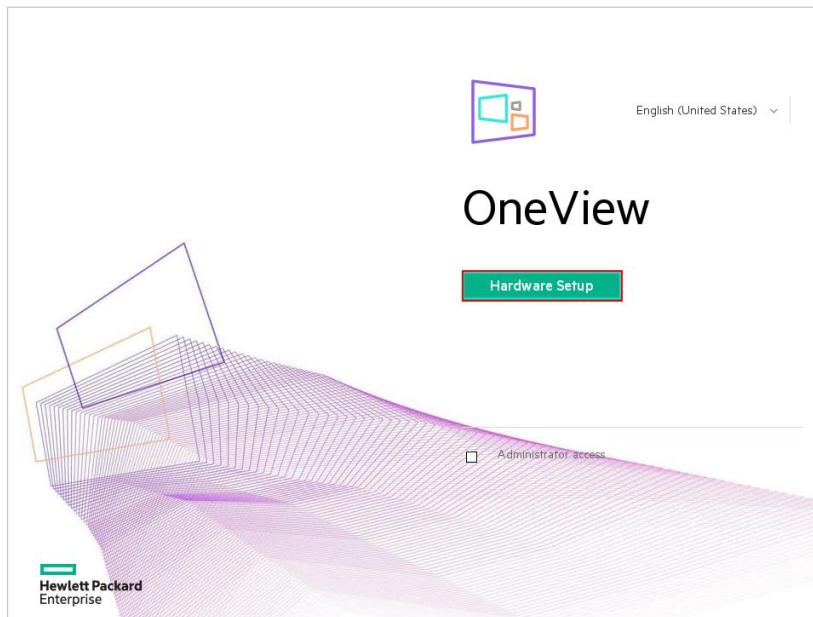


## Chapter-4 –Initial Configuration of the Demonstration Appliance

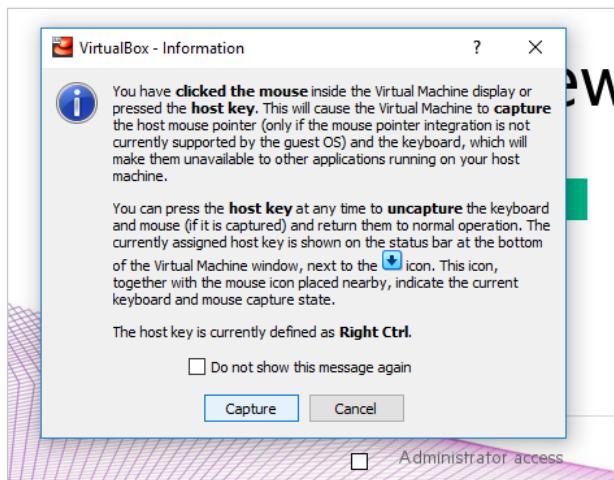
We have completed the installation of our PowerShell/Python scripting languages, prepared the Ansible environment with the required modules and our demonstration appliance must have finished booting up by now, so now we can finalize the configuration of our appliance.

### Hardware discovery and initial appliance configuration

- Access to the console and press **Hardware Setup**



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





# Hewlett Packard Enterprise

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

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

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

Windows IP Configuration

Ethernet adapter Ethernet0:

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

Ethernet adapter VirtualBox Host-Only Network:

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

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

**IPv4**

Address assignment  None  Manual

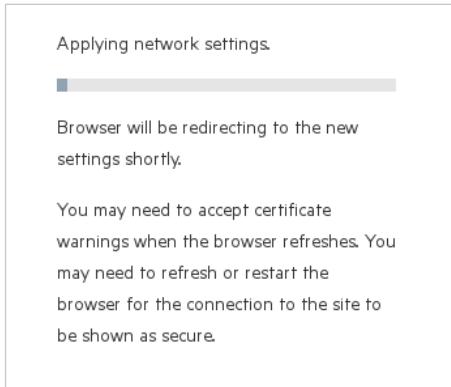
Once IPv4 is disabled, it cannot be re-enabled without reinstalling the appliance. See the OneView Install guide for supported configurations when using only IPv6 addresses.  
[Learn more...](#)

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



# Hewlett Packard Enterprise

- Then click **OK**



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

HPOneView-DCS\_5.00.00-0400525 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

OneView Search

Actions

Hardware Setup

Checklist 1

**⚠ Setup incomplete**

**▼ ▲ Hardware is being discovered** 1/14/20 4:39:53 pm  
1 minute 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.

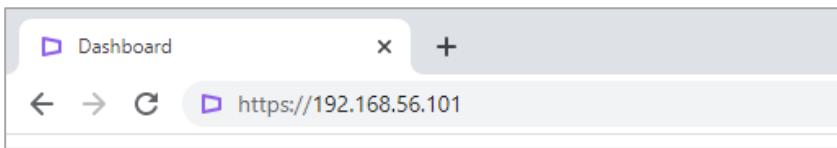
Inventory 6

Add Running

0 of 3 enclosures added  
0000A66102, interconnect 1:  
Check for the interconnect responsiveness.

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

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





# Hewlett Packard Enterprise

- Accept the OneView license agreement

OneView License

Your use of the HPE OneView software is subject to the Hewlett Packard Enterprise End User License Agreement ("EULA") below, and any additional terms described in the Additional License Authorization ("ALA") for HPE OneView that can be found at <http://www.hpe.com/info/hpeoneview/eula>. For the information regarding HPE privacy practices, please refer to HPE Privacy Statement that can be found at: <https://www.hpe.com/us/en/privacy/ww-privacy-statement.html>.

If you agree with the licensing terms, click Agree. Otherwise, click Disagree.

Hewlett Packard Enterprise End User License Agreement - Enterprise Version (v1.0 2017)

1. **Applicability.** This end user license agreement (the "Agreement") governs the use of accompanying software, unless it is subject to a separate agreement between you and Hewlett Packard Enterprise Company and its subsidiaries ("HPE"). By downloading, copying, or using the software you agree to this Agreement. HPE provides translations of this Agreement in certain languages other than English, which may be found at: <http://www.hpe.com/software/SWLicensing>.

2. **Terms.** This Agreement includes supporting material accompanying the software or referenced by HPE, which may be software license information, additional license authorizations, software specifications, published warranties, supplier terms, open source software licenses and similar content ("Supporting Material"). Additional license authorizations are at: <http://www.hpe.com/software/SWLicensing>.

3. **Authorization.** If you agree to this Agreement on behalf of another person or entity, you warrant you have authority to do so.

4. **Consumer Rights.** If you obtained software as a consumer, nothing in this Agreement affects your statutory rights.

5. **Electronic Delivery.** HPE may elect to deliver software and related software product or license information by electronic transmission or download.

6. **License Grant.** If you abide by this Agreement, HPE grants you a non-exclusive non-transferable license to

If you agree with the licensing terms, click Agree. Otherwise, click Disagree.

**Agree** **Disagree**

- Leave OneView Support enabled and select OK

OneView Support

Application Support

This product contains a technical feature that will allow HPE support personnel to access your system, through the system console, to assess problems that you have reported. This access will be controlled by a password generated by HPE that will only be provided to authorized support personnel. You can disable access at any time while the system is running.

For additional information on Support Access, see the HPE OneView User Guide.

Authorized services access **Enabled**

HPE Open Source Download Site

Open source software license information and source for HPE OneView can be obtained at <http://www.hpe.com/software/opensource>

**OK**



# Hewlett Packard Enterprise

- Log in using **Administrator / admin**

The image shows a screenshot of the OneView login interface. It features a header "OneView" and two input fields: one for "User" containing "Administrator" and another for "Password" containing "\*\*\*\*\*". Below these is a green "Login" button.

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

The image shows a "OneView" dialog box titled "Assign an administrator password." It contains fields for "User" (set to "Administrator"), "New password" (containing "\*\*\*\*\*"), and "Confirm password" (also containing "\*\*\*\*\*"). A green "OK" button is at the bottom.

- Verify the demonstration appliance is ready to be configured. When the initial hardware discovery is completed you should see:

The image shows a sidebar menu with an "Actions" dropdown. Under "Inventory 45", there are links for "Enclosures 3 >", "Server Hardware 21 >", "Drive Enclosures 3 >", and "Interconnects 18 >".

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

**Note:** Don't wait for the discovery to complete as it takes about 15-20mn, go to **Chapter-4** and start the installation/configuration of Postman. When the discovery is complete, you can continue Chapter-3.



# Hewlett Packard Enterprise

## Creation of an initial setup snapshot

The initial appliance configuration is completed, we can create a snapshot so that we have the initial configuration saved.

**Note:** Before creating the snapshot:

- Make sure the initial hardware discovery is completed

OneView

Search

Actions

Hardware Setup

Checklist 0

Hardware discovery complete

Inventory 45

Endlosures 3 >

Server Hardware 21 >

Drive Enclosures 3 >

Interconnects 18 >

- 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:

OneView

state:running

Actions

Activity 0

Name Resource Date State Owner

No matches

Pending

Running 2

Completed

Interrupted

Error

Warning

Suspended

Cancelling

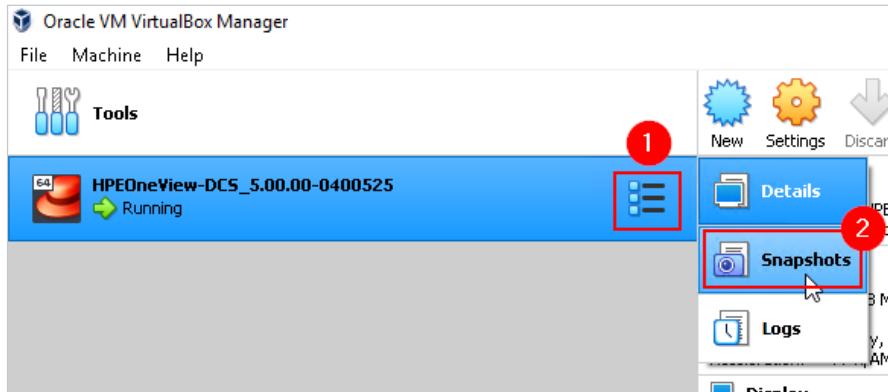
**Note:** Taking a snapshot while the appliance is running is a key practice to get the appliance up and running almost instantaneously.



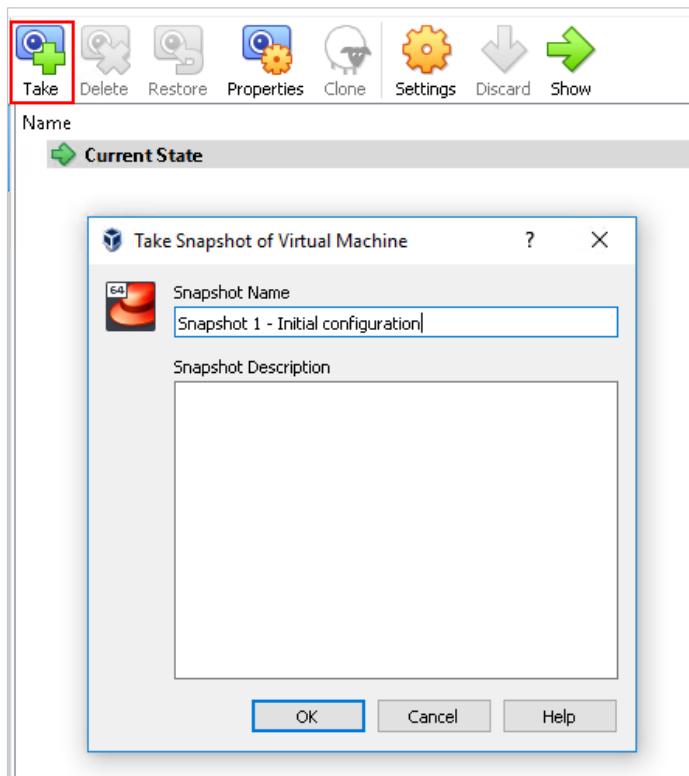
# Hewlett Packard Enterprise

To create a snapshot with the initial configuration:

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



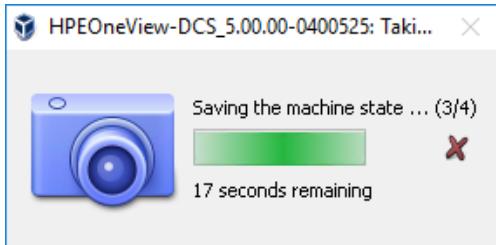
- Then press on **Take**, enter a snapshot name like **Snapshot 1 - Initial configuration** then click **OK**





# Hewlett Packard Enterprise

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



This concludes Chapter-4

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



## Chapter-5 – Preparing Postman

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

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

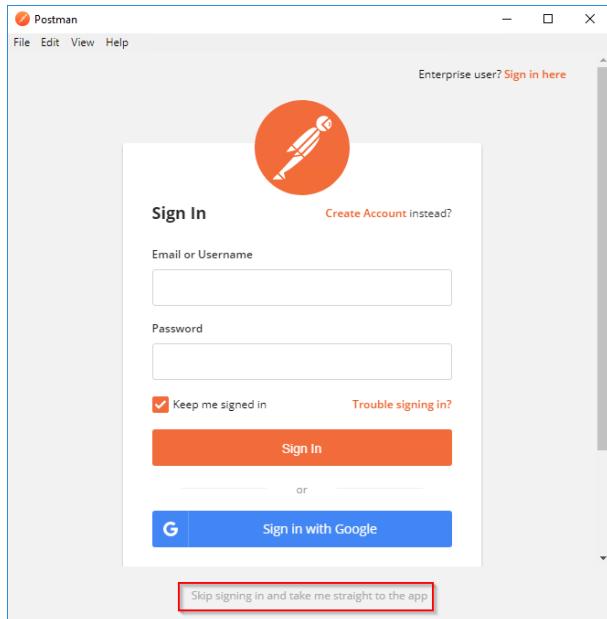
There are several simple solutions for this. My favorite is Postman as it is one of the most complete REST tools and offers useful features:

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

## Installing and configuring Postman

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

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

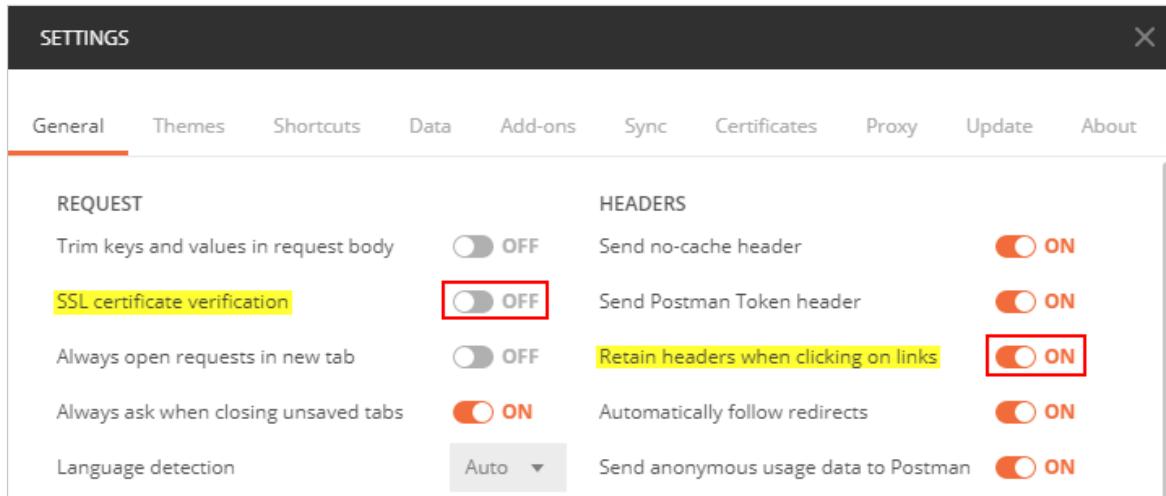




# Hewlett Packard Enterprise

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.



# Hewlett Packard Enterprise

- Then click **Raw**

1995 lines (1995 sloc) | 43.6 KB

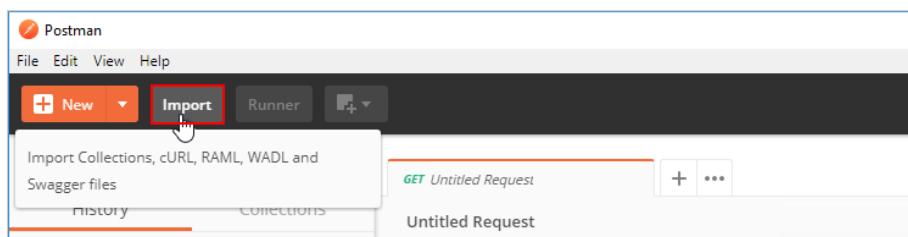
```
1 {
2     "info": {
3         "_postman_id": "525f7596-a659-556e-ea26-332682d540d2",
4         "name": "OneView",
5         "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
6     },
}
```

Raw Blame History

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



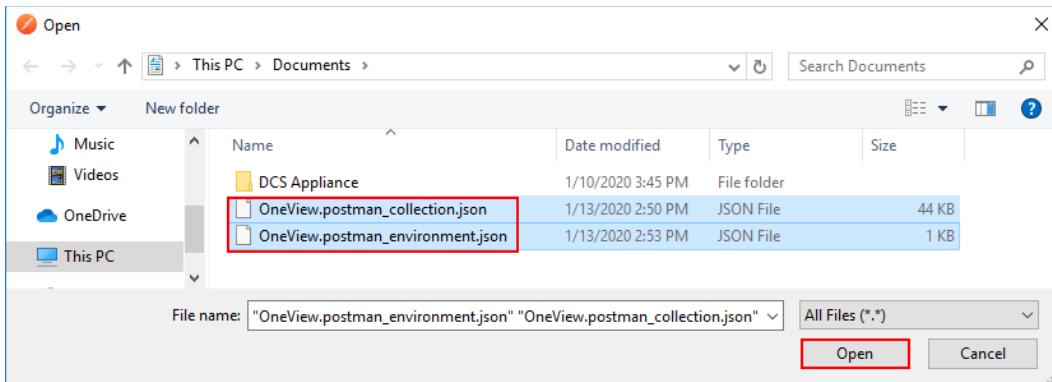
- Do the same for **OneView.postman\_environment.json**.
- Back in Postman, click on **Import**:





# Hewlett Packard Enterprise

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



- You should see in collections, the new OneView collection

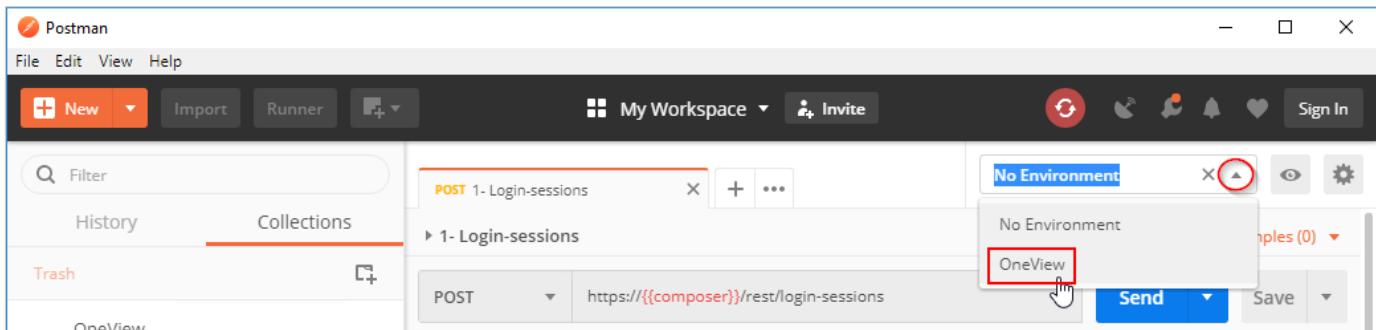
A screenshot of the Postman application interface. On the left, there is a sidebar with tabs for 'History', 'Collections' (which is highlighted with a red box), and 'Trash'. Below these are sections for 'OneView' (49 requests) and other categories like 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. On the right, the main panel displays a 'GET Untitled Request' with various request details such as method (GET), URL, parameters (Params and Authorization), and response preview. A list of requests under 'OneView' is shown on the right side of the main panel.



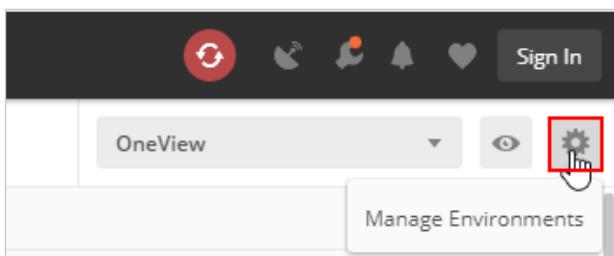
## Configuration of the Postman environment

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.

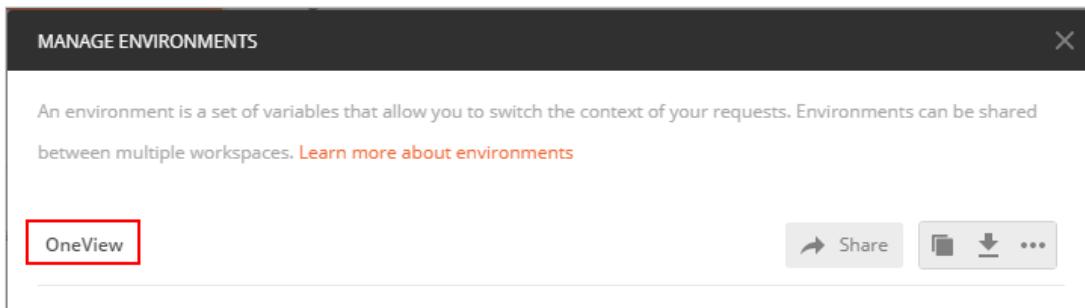
- Expand the environment menu and select **OneView**



- Then select **Manage Environments**



- Click on **OneView**





# Hewlett Packard Enterprise

There are two variables defined as illustrated below:

MANAGE ENVIRONMENTS

Environment Name  
OneView

	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**

MANAGE ENVIRONMENTS

Environment Name  
OneView

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	composer	composer.lj.lab	192.168.56.101 <span style="border: 1px solid red; padding: 2px;">1</span>	X	...	
<input checked="" type="checkbox"/>	xapiversion	1200	1200			
	Add a new variable					

3

Cancel 2 Update

# Hewlett Packard Enterprise

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

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', 'OneView' is expanded, showing 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. Below these is a red box around 'POST 1- Login-sessions'. The main workspace shows a 'POST 1- Login-sessions' request with the URL `https://{{composer}}/rest/login-sessions`. The 'Body' tab is selected, showing a table with a single row: 'Key' and 'Value'. The 'Send' button is highlighted with a blue box.

- 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 of the Postman interface shows the same setup as the previous one, but with several annotations: a red circle labeled '1' points to the 'Body' tab in the request settings; a red box labeled '2' highlights the JSON body content: 

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

; and a red circle labeled '3' points to the 'Save' button, which is also highlighted with a red box.

- Then press **Send**



# Hewlett Packard Enterprise

The response we should get is the following:

The screenshot shows the Postman interface with a successful response. The status bar at the top indicates "Status: 200 OK". The body section contains the following JSON response:

```

1  {
2     "sessionID": "LTIyNjY0MTk3Mzkxz2ain_AKtwH9jH1PSVOnW3EB4EQQqnkT",
3     "partnerData": {}
4 }

```

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:

The screenshot shows the "Tests" tab for a POST request to "https://{{composer}}/rest/login-sessions". The test script contains the following code:

```

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

```

A note below the script states: "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:

The screenshot shows the "Headers" tab for a GET request to "https://{{composer}}/rest/version". The headers are defined as follows:

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

The status bar at the bottom indicates "Status: 200 OK".



# Hewlett Packard Enterprise

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 ▾

Author	Commit Message	Created
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

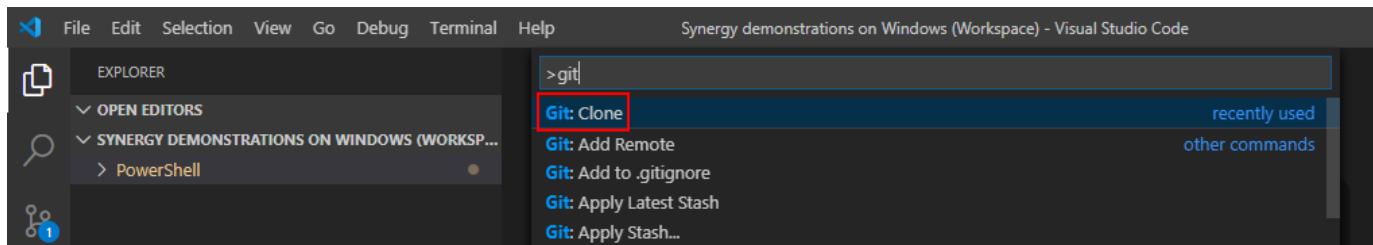


# Hewlett Packard Enterprise

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.

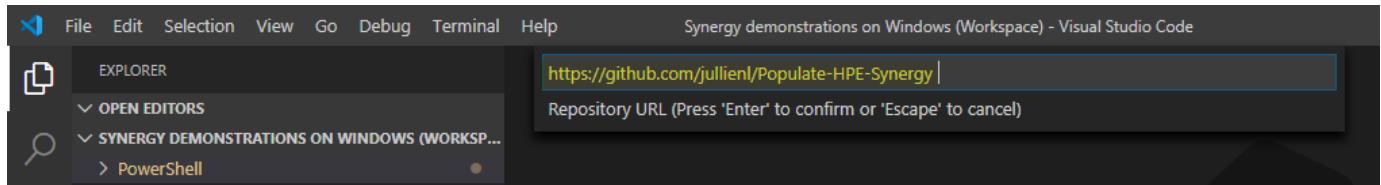
To clone the Dave's repo:

- 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:

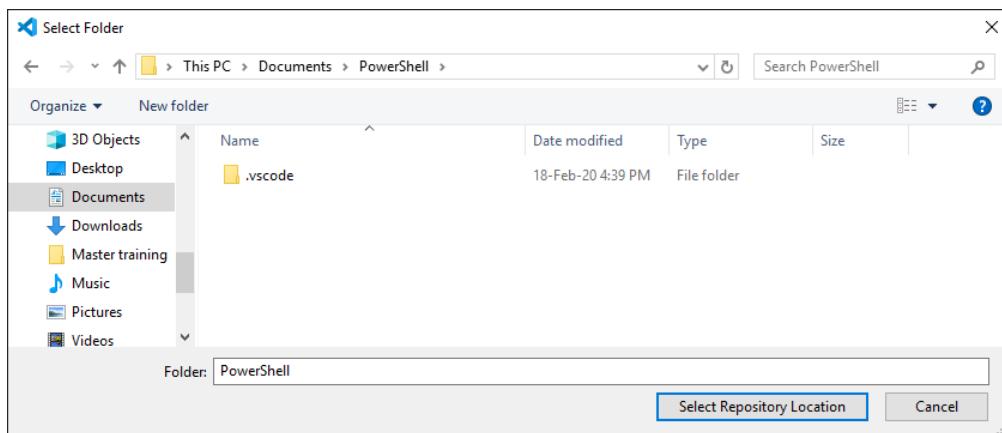


- Enter the following link: <https://github.com/jullienl/Populate-HPE-Synergy>

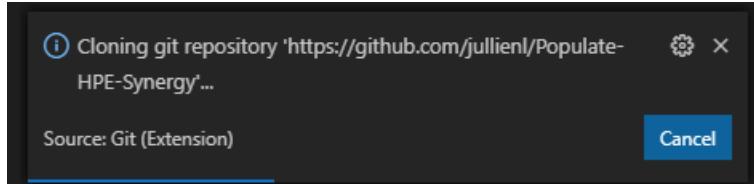
**Note:** This repository is forked from daveolker/Populate-HPE-Synergy. This fork provides a few changes from Dave's repository to meet the needs of our demonstration scenarios.



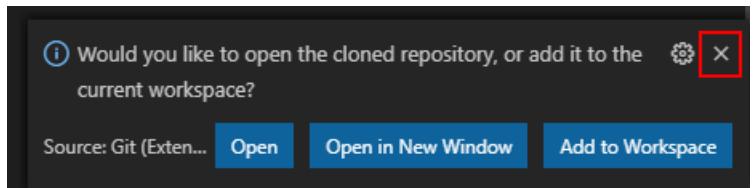
- Then select your Windows user home workspace folder as the parent directory under which to put the Dave's forked repository:



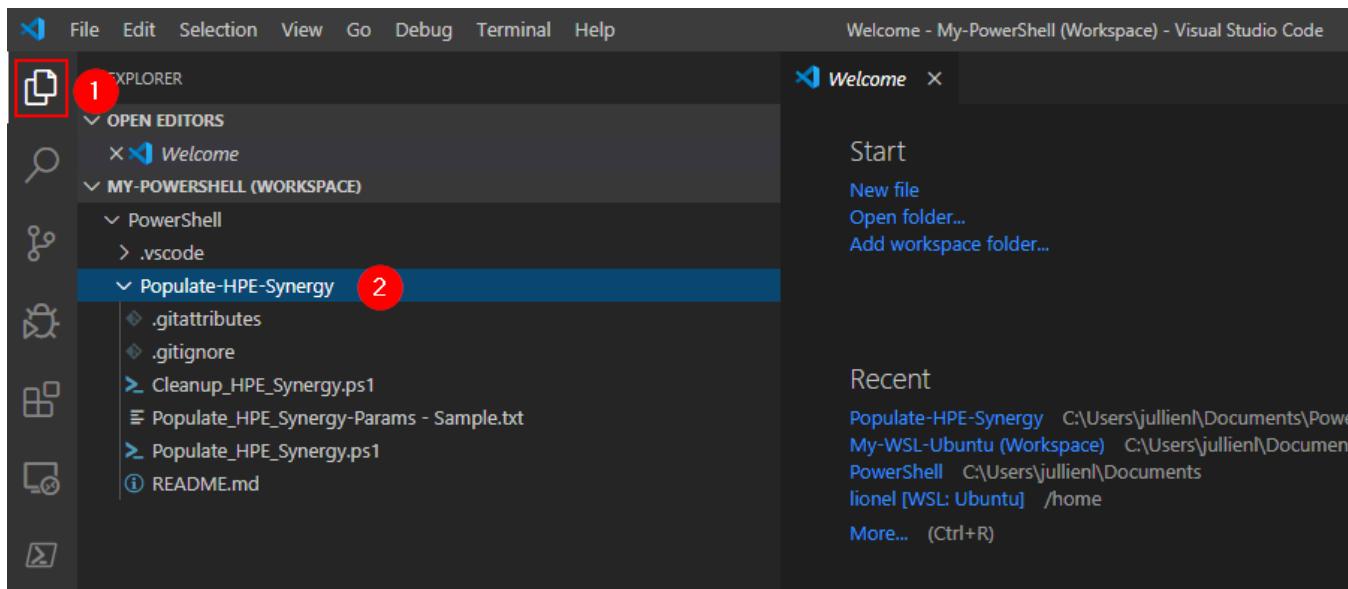
- Once selected, the cloning starts:



- 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:

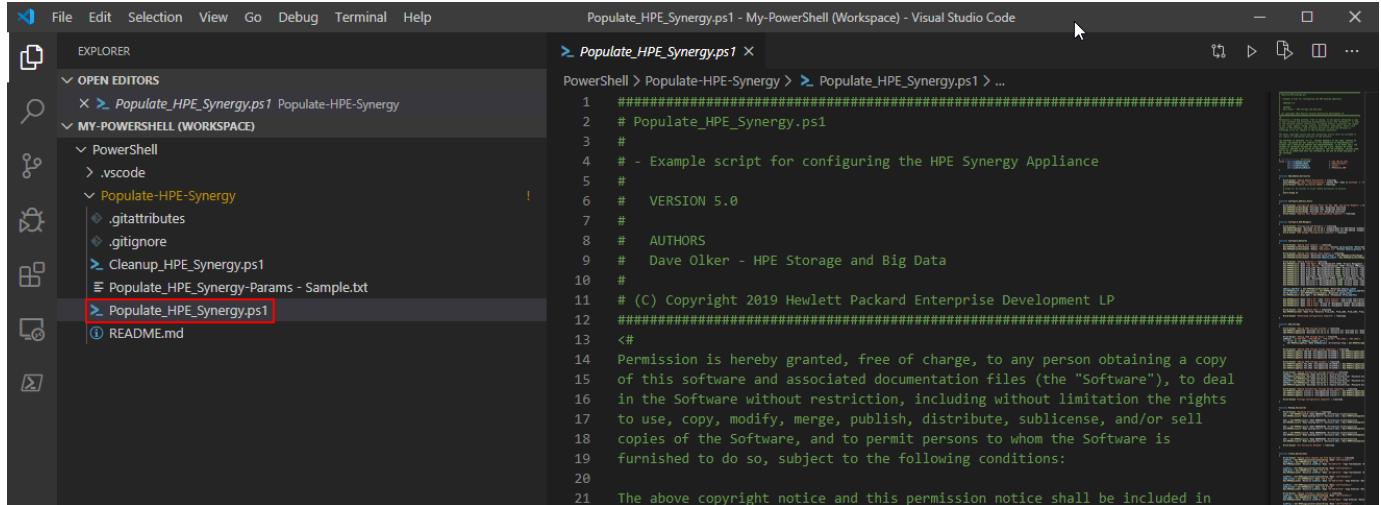
`Populate_HPE_Synergy.ps1` script connects with the appliance and discovers/configures all the simulated hardware.

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.



# Hewlett Packard Enterprise

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



```
> 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

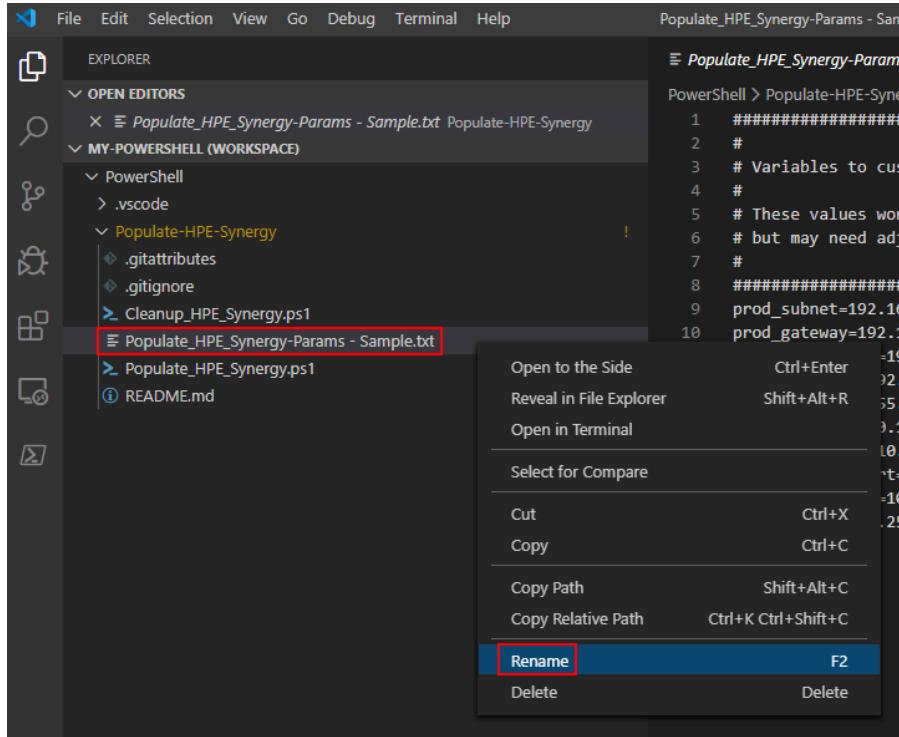
**Note:** To get a better display of the script, press **CTRL + K + CTRL + O** to collapse all regions



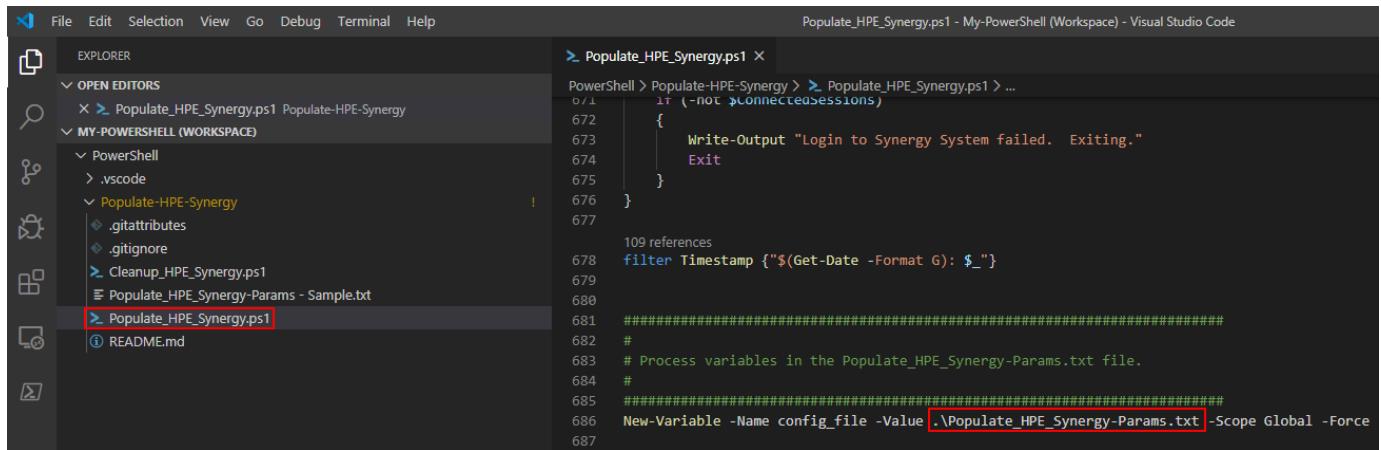
# Hewlett Packard Enterprise

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_Template_SY480_Gen9_RHEL_Local_Boot
722 Create_Server_Profile_Template_SY660_Gen9_Windows_SAN_Storage
723 Create_Server_Profile_Template_SY480_Gen9_ESX_SAN_Boot
724 Create_Server_Profile_Template_SY480_Gen10_ESX_SAN_Boot
725
```

# Hewlett Packard Enterprise

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.

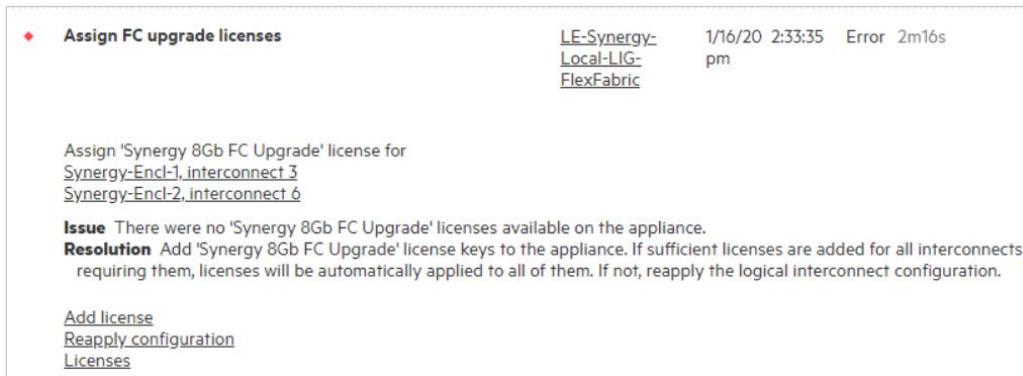
**Note:** You can download an HPE Synergy SPP from <https://www.hpe.com/downloads/synergy>

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

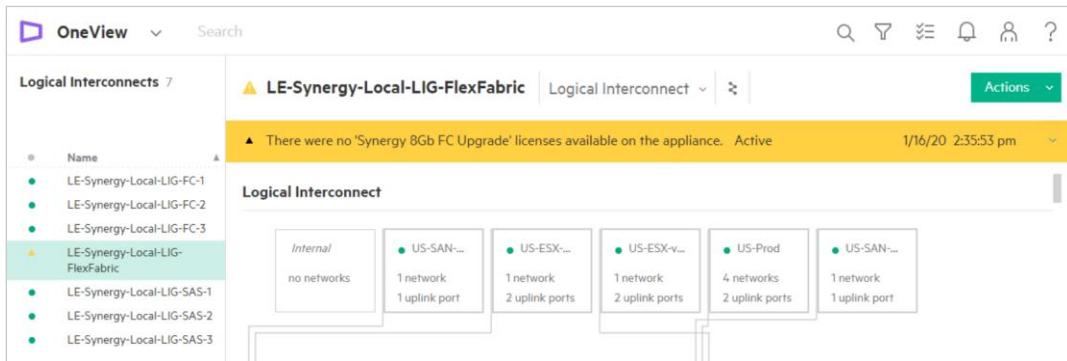
When run, the function asks to specify the location of a Service Pack for ProLiant ISO file.

- `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:

- Throwing an error during the creation of the Logical Enclosure:



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

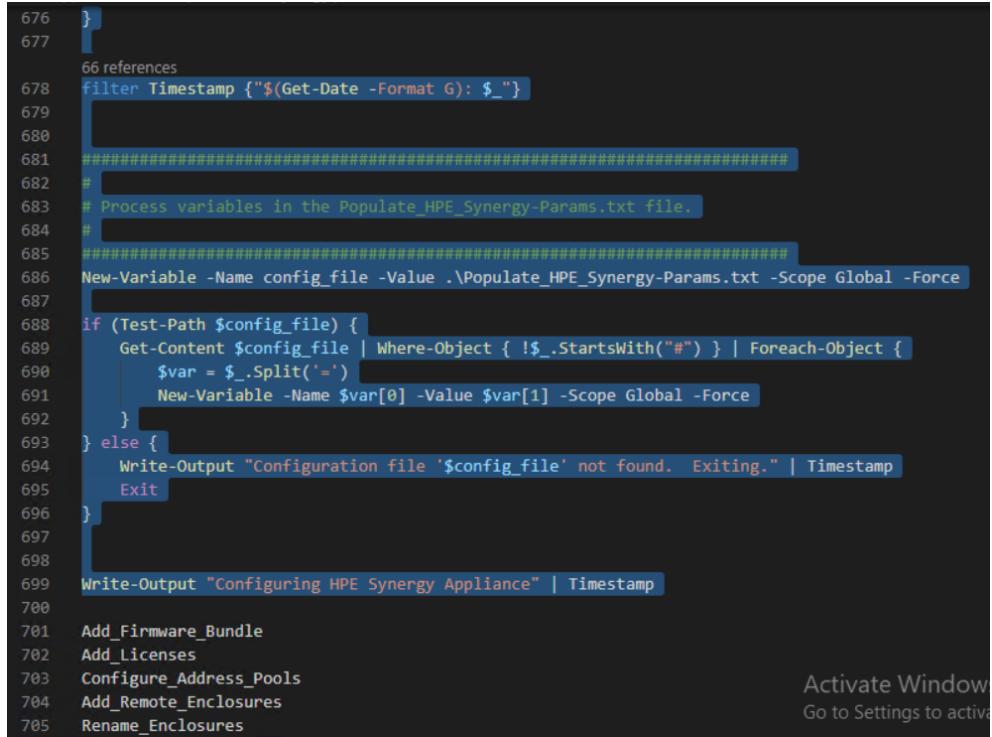




# Hewlett Packard Enterprise

- `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

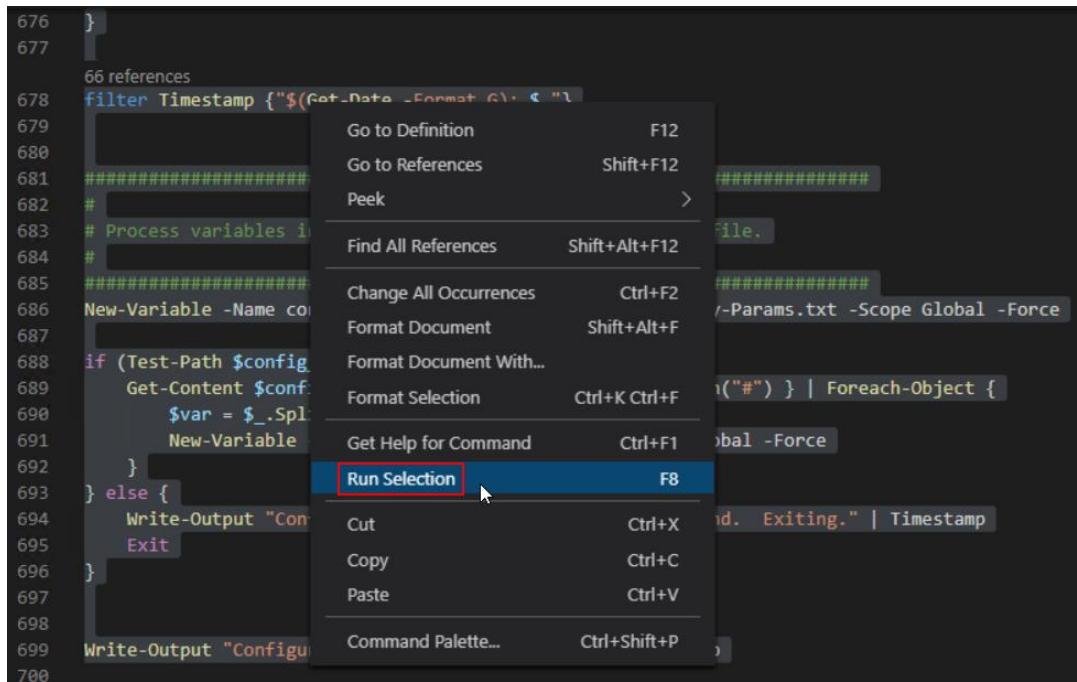
- Now select the first part of the script from line 1 to 699



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

Activate Window  
Go to Settings to activate

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



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



# Hewlett Packard Enterprise

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: PowerShell Integrate + ⌂ ⌂ ⌂ ⌂

```
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> |

Activate Windows  
Go to Settings to activate Windows.

Ln 700, Col 1 (38425 selected) Spaces: 4 UTF-8 LF PowerShell ☐ 5.1 ⌂ ⌂ ⌂ ⌂

- Make sure no error is thrown.





# Hewlett Packard Enterprise

- 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 terminal window in VS Code with the following content:

```
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
```

Below the code, there are tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected. The terminal output shows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Powershell

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> 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`, the SPP ISO file is optional, you can press **ENTER**
- For `Add_License`, the licenses are optional, you can press **ENTER**

**Note:** `Add_Remote_Enclosures` takes some time but is optional so for the sake of time, we will skip it!

- 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
```



# Hewlett Packard Enterprise

## Creation of a final setup snapshot

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

**Note:** Before creating the snapshot:

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

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

The screenshot shows the HPE OneView interface. On the left, there's a navigation pane with a 'Logical Enclosures' section containing a '+ Create logical enclosure' button. A tree view shows 'Name' and 'LE-Synergy-Local'. The main panel displays a logical enclosure named 'LE-Synergy-Local' with an 'Overview' tab selected. A yellow warning banner at the top states: '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](#)'. It lists several components: 'EG-Synergy-Local', 'Synergy-Encl-1', 'Synergy-Encl-2', 'Synergy-Encl-3', and various logical interconnects like 'LE-Synergy-Local-LIG-FlexFabric' through 'LE-Synergy-Local-LIG-SAS-3'.

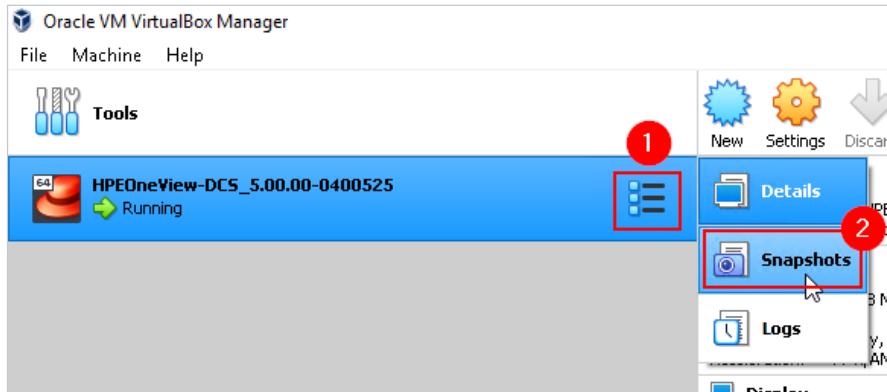
- Make sure there is no OneView tasks that is still running
  - Go in OneView / **Activity** then use **Running** State in the Filter tool:

The screenshot shows the 'Activity' filter tool in HPE OneView. The search bar contains 'state:running'. A red circle with the number '1' is on the search icon. A red box highlights the 'Running' option in the filter dropdown menu, which has a red circle with the number '2' on it. The dropdown also includes other states: Pending, Completed, Interrupted, Error, Warning, Suspended, and Cancelling. The main table below shows a single row with 'No matches'.

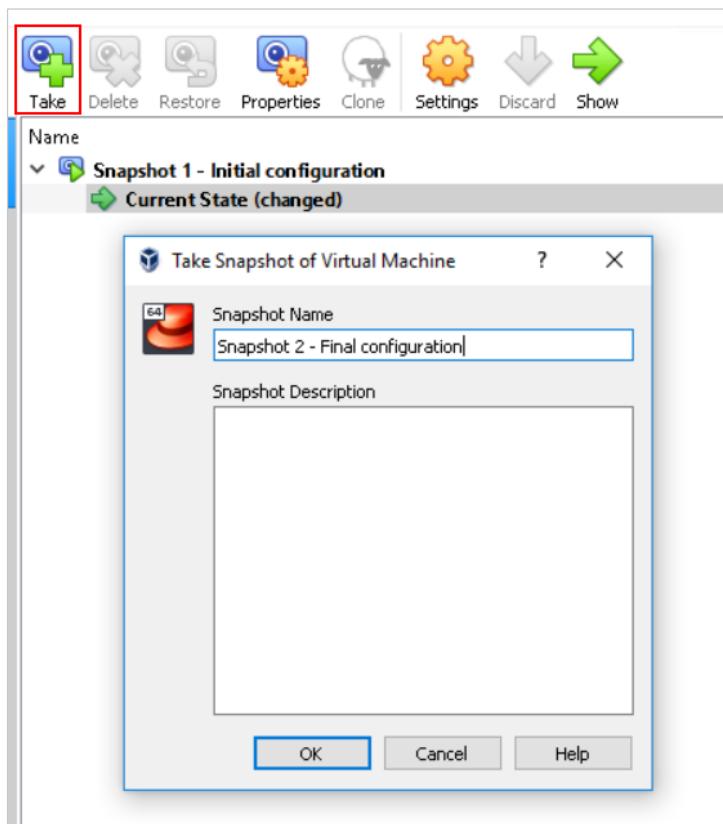
**Note:** Taking a snapshot while the appliance is running is a key practice to avoid waiting long minutes for the appliance to start.

To create a snapshot to save the appliance final configuration:

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



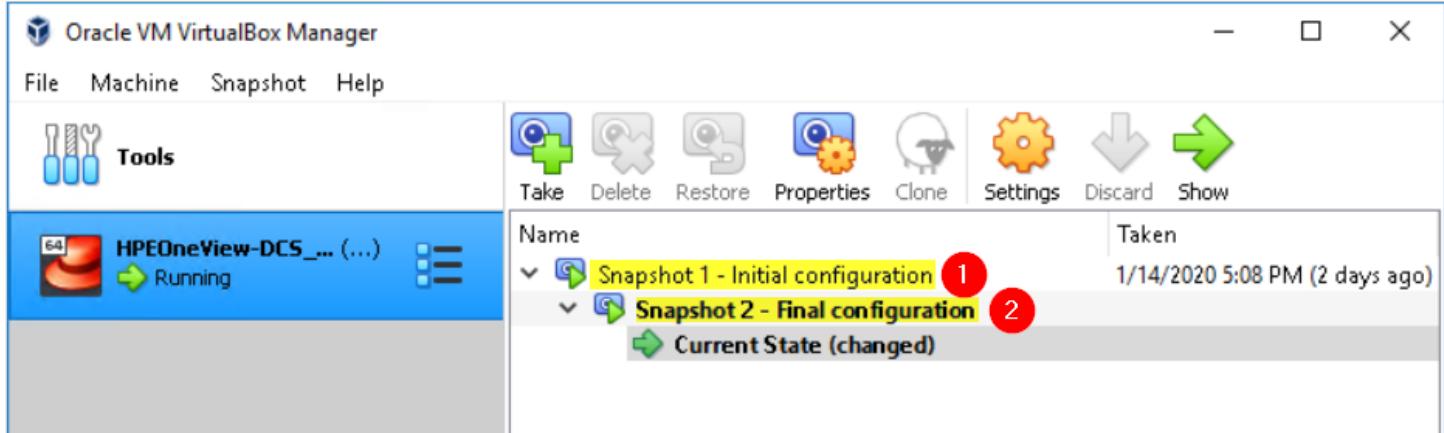
- Then press on **Take**, enter a snapshot name like **Snapshot 2 - Final configuration** then click **OK**





# Hewlett Packard Enterprise

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

More scenarios will be added over time...



# Hewlett Packard Enterprise

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

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

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

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

The screenshot shows the Postman application window. The left sidebar lists collections under 'OneView' with 49 requests, including 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. The main pane displays a collection named '1- Login-sessions' with a POST request. The request details are as follows:

- Method:** POST
- URL:** `https://{{composer}}/rest/login-sessions`
- Params:** None
- Headers:** (2)
- Body:** (1) JSON

```
1 [ {  
2   "sessionID": "LTY2NjcxMjAxMDI1dyKaBMDieS-ma8FgOXPhHgCtSoeeSnt0",  
3   "partnerData": {}  
4 } ]
```
- Tests:** None
- Cookies:** None
- Code:** None
- Comments:** (0)

Annotations with red circles and numbers:

- Annotation 1: Points to the 'Body' tab in the request details panel.
- Annotation 2: Points to the 'Send' button in the top right of the request details panel.
- Annotation 3: Points to the 'Body' tab in the bottom navigation bar of the request details panel.

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



# Hewlett Packard Enterprise

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

The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view under 'OneView' containing 'Logs', 'Reserved VLAN range', 'eFuse', and 'Associations'. Below this is a list of API requests:

- POST** 1- Login-sessions
  - GET** 2- Get-X-API-Version
  - GET** Profile Templates
  - GET** Get-Ethernet-Networks
  - GET** Get-Datacenters
  - GET** Get the OS Deployment plans from ...
  - GET** Get the Hypervisor Managers from ...
  - GET** Get the Hypervisor Cluster Profiles ...
  - PUT** Upload CRL
  - GET** Get-Interconnect **1** 
  - GET** Get-interconnect VC40G Frame1-ba...
  - GET** Get-Interconnect VC40G Frame1-Ba...

At the top right, there are buttons for 'Send' (highlighted with a red box and circle '2'), 'Save', and other options. Below the 'Send' button is a status bar showing 'Status: 200 OK'.

- The response displays all interconnects managed by OneView



# Hewlett Packard Enterprise

- 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**

POST 1-Login-session    GET Get-Interconnect    GET https://{{con...}}    + ...    OneView

https://{{composer}}/rest/enclosures/000000000A66103

GET https://{{composer}}/rest/enclosures/000000000A66103    Send

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

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Response

- The response provides information about the enclosure UUID 000000000A66103



# Hewlett Packard Enterprise

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

Body Cookies Headers (13) Test Results Status: 200 OK Time: 163 ms Size: 19.34 KB Download

Pretty Raw Preview JSON

```

5   "elag": "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": "00000000A66103",
14  "serialNumber": "0000A66103",
15  "partNumber": "000000-010",
16  "reconfigurationState": "NotReapplyingConfiguration",
17  "uidState": "On",
18  "licensingIntent": "NotApplicable",
19  "deviceBayCount": 12,
20  "deviceBays": [ ],
360  "interconnectBayCount": 6,
361  "interconnectBays": [ ],
453  "fanBayCount": 10,
454  "fanBays": [ ],
576  "powerSupplyBayCount": 6,
577  "powerSupplyBays": [ ],
651  "enclosureGroupUri": null,
652  "fwBaselineUri": null,
653  "fwBaselineName": null,
654  "isFwManaged": false,
655  "forceInstallFirmware": false,
656  "logicalEnclosureUri": null,
657  "managerBays": [
658    {
659      "bayNumber": 1,
660      "managerType": "EnclosureManager",
661      "uidstate": "Off",
662      "bayPowerState": "Unknown",

```

**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

OneView Search

Dashboard Server Profiles 0 > Server Hardware 21 >

No server profiles

4 Warning

Help

- Screencasts
- Tutorial
- Documentation
- Help on this page
- Browse help
- 2 REST API reference**
- SDK & partner program
- First time setup
- License

# Hewlett Packard Enterprise

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

The screenshot shows the HPE OneView API Reference interface. On the left, there's a sidebar with various navigation links. A red circle labeled '1' highlights the 'Servers' section. A red circle labeled '2' highlights the 'Enclosures' link under the 'Servers' section. A red circle labeled '3' highlights the 'GET /rest/enclosures' method. The main content area is titled 'Enclosures' and describes the resource for managing enclosures. It includes a 'Request' section with a code snippet:

```
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

The screenshot shows the 'Response Body' section for the 'Enclosures' resource. It lists several components with their types and descriptions. Red boxes highlight specific fields: 'applianceBayCount', 'applianceBays', 'bayNumber', 'bayPowerState', 'Values', 'EFuse', and 'Reset'. The descriptions for these highlighted fields are:

- 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.
- Values**: The power state of the bay has been EFused.
- EFuse**: The power state of the bay has been reset.
- Reset**: The power state of the bay has been reset.



# Hewlett Packard Enterprise

- 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, the 'Collections' view is open, showing various resources like Logs, Reserved VLAN range, and eFuse. The 'Reserved VLAN range' item is selected and highlighted with a red circle labeled '1'. Within this section, a specific GET request labeled '3- Get-Reserved-VLAN-range' is selected and highlighted with a red circle labeled '2'. On the right, the main workspace displays a JSON response for a GET request to '/rest/fabrics/'. A red box highlights the 'Send' button at the top right of the workspace area, and another red circle labeled '3' points to it. The JSON response shows a 'members' array containing one fabric object, which includes a 'reservedVlanRange' field. The 'start' value is highlighted in yellow as '3967' and the 'length' value is highlighted in green as '128'.

```
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-6841-4835-85b2-27bfafe0d3bc8",
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"
40  }
41 ]
```

**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, the 'OneView' collection is expanded, showing several requests including '1- Login-sessions', '2- Get-X-API-Version', '3- Get-Reserved-VLAN-range', '4- Put-Change-Reserved-VLAN-range' (which is highlighted with a red box and labeled '1'), and '5- Get-Task-object-Result'. On the right, the '4- Put-Change-Reserved-VLAN-range' request is selected. The 'Body' tab is highlighted with a red box and labeled '2'. The JSON code in the body 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 '4- Put-Change-Reserved-VLAN-range' request. The 'Headers' tab is highlighted with a red box. The 'Status' bar at the top shows '202 Accepted'. The 'Headers' section contains the following entries:

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

# Hewlett Packard Enterprise

- 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 requests. One request, '5- Get-Task-object-Result', is highlighted with a red box and a circled '1'. On the right, the details for this request are shown: method 'GET', URL '{{taskresult}}', and the 'Send' button highlighted with a red box and a circled '2'. The response body is displayed in JSON format:

```
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

# Hewlett Packard Enterprise

- 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 red circle labeled '1' highlights the '6- Get-New-Reserved-VLAN-range' request under the 'OneView' collection. On the right, the main window displays the details for this request. The 'Send' button, located at the top right of the request details panel, is highlighted with a red box and a red circle labeled '2'. The request URL is `https://{{composer}}/{{fabricid}}/reserved-vlan-range`. The response body is shown 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



# Hewlett Packard Enterprise

## 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**

The screenshot shows the OneView interface. On the left, the navigation bar has 'Dashboard' selected. Under 'SERVERS', 'Server Profiles' is selected and highlighted with a red box and a cursor icon. Other options include 'Server Profile Templates', 'Enclosure Groups', and 'Logical Enclosures'. The main dashboard shows 'Server Profiles 0 >' and 'No server profiles'. To the right, 'Server Hardware 39 >' is shown with a large green circle highlighting the '39 OK' status indicator. The top right of the screen includes a search bar, filter icons, and user profile icons.

- Click on **Create profile**



# Hewlett Packard Enterprise

- Name the profile **Profile-1**, use the server hardware **Synergy-Encl-1, bay 7** with the Server Profile Template **HPE Synergy 480 Gen9 with Local Boot for RHEL Template**:

**Create Server Profile** General ?

**General**

Name	Profile-1
Scope	Select zero or more scopes <input type="button" value=""/>
Server profile template	HPE Synergy 480 Gen9 with Local Boot for RHEL Template <input type="button" value=""/>
Description	Server Profile for HPE Synergy 480 Gen9 Compute Mod
Server hardware	Synergy-Encl-1, bay 7 <input type="button" value=""/>
<input type="checkbox"/> Show empty bays	
Server hardware type	SY 480 Gen9 1
Enclosure group	EG-Synergy-Local
Affinity	Device bay <input type="button" value=""/>

**OS Deployment**

OS deployment plan	None <input type="button" value=""/>
--------------------	--------------------------------------

**Firmware**

Firmware baseline	managed manually <input type="button" value=""/>
-------------------	--

 Changed: Server hardware to "Synergy-Encl-1, bay 7" **Create** **Create +** **Cancel**

- 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.



# Hewlett Packard Enterprise

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

**Edit Profile-1** General ▾

**General**

Name	Profile-1
Description	Server Profile for HPE Synergy 480 Gen9 Compute Mod
→ Server profile template	HPE Synergy 480 Gen9 with Local Boot for RHEL Template <a href="#">Change</a>
Server hardware	Synergy-Encl-1, bay 7 <a href="#">X</a> <a href="#">🔍</a>
<input type="checkbox"/> Show empty bays	
Server hardware type	SY 480 Gen9 1 <a href="#">Change</a>
Enclosure group	EG-Synergy-Local <a href="#">Change</a>
Affinity	Device bay ▾

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

**Change Server Profile Template** ?

Changing the server profile template may require private volume attachments on the server profile to be re-associated with private volume attachments on the newly specified server profile template. [Learn more](#)

Server profile template	None <a href="#">X</a> <a href="#">🔍</a>
-------------------------	--

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



# Hewlett Packard Enterprise

- 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:

The screenshot shows a Windows PowerShell window with the title bar '3: powershell'. The terminal tab is selected. The command history and output are as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\jullienl\Documents\PowerShell> #IP address of OneView
PS C:\Users\jullienl\Documents\PowerShell> $IP = "192.168.56.101"
PS C:\Users\jullienl\Documents\PowerShell> # OneView Credentials
PS C:\Users\jullienl\Documents\PowerShell> $username = "Administrator"
PS C:\Users\jullienl\Documents\PowerShell> $password = "password"
PS C:\Users\jullienl\Documents\PowerShell>
PS C:\Users\jullienl\Documents\PowerShell> $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
PS C:\Users\jullienl\Documents\PowerShell> $credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
PS C:\Users\jullienl\Documents\PowerShell>
PS C:\Users\jullienl\Documents\PowerShell> Connect-HPOVMgmt -appliance $IP -Credential $credentials

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

PS C:\Users\jullienl\Documents\PowerShell>
```

- 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
```

The screenshot shows a Windows PowerShell window with the title bar '3: powershell'. The terminal tab is selected. The command history and output are as follows:

```
PS C:\Users\jullienl\Documents\PowerShell> get-hpovserverprofile -name Profile-1

Name      Status Compliance Template                               Server Hardware      Server Hardware Type Enclosure Group
----      ----  -----          -----
Profile-1 OK    Compliant HPE Synergy 480 Gen9 with Local Boot for RHEL Template Synergy-Encl-1, bay 7 SY 480 Gen9 1   EG-Synergy-Local

PS C:\Users\jullienl\Documents\PowerShell>
```



# Hewlett Packard Enterprise

- 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
```

```
PS C:\Users\Administrator.lj\Documents\DCS Appliance> Get-HPOVServerProfile -Name Profile-1 | ConvertTo-HPOVPowerShellScript
# ----- Attributes for ServerProfile "Profile-1"
$Name          = "Profile-1"
$Server        = Get-HPOVServer -Name "Synergy-Encl-1, bay 11"
$Affinity      = "Bay"
# ----- Attributes for connection "1"
$connID        = 1
$connType      = "Ethernet"
$netName       = "ESX Mgmt"
$thisNetwork   = Get-HPOVNetwork -Type Ethernet -Name $netName
$portID        = "Mezz 3:1-a"
$requestedMbps = 2500
$conn1         = New-HPOVServerProfileConnection -ConnectionID $connID -ConnectionType $connType -Network $thisNetwork
$rtID          = RequestedBW $requestedMbps
# ----- Attributes for connection "2"
$connID        = 2
$connType      = "Ethernet"
$netName       = "ESX Mgmt"
$thisNetwork   = Get-HPOVNetwork -Type Ethernet -Name $netName
$portID        = "Mezz 3:2-a"
$requestedMbps = 2500
```

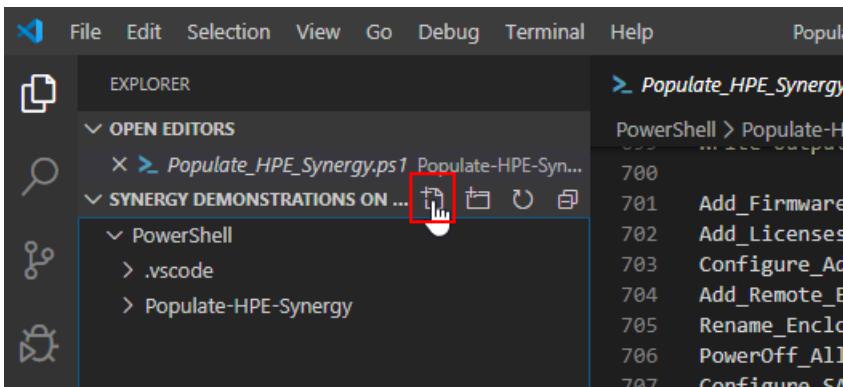
Explain to the customer that this cmdlet can assist administrators or scripters to help generate script code from specific resources. The code generated is a starting point to be used for repeating similar tasks performed by the UI, or to incorporate into scripts or workflows.

**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, to disconnect properly the PowerShell client to the appliance, enter:

```
Disconnect-HPOVMgmt
```

- You can create now a new file using **New File**





# Hewlett Packard Enterprise

- Name it **Profile-2-creation.ps1**. 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
- Title Bar:** Profile-2-creation.ps1 - My-PowerShell (Workspace) - Visual Studio Code
- Explorer Panel:** Shows 1 UNSAVED file named Profile-2-creation.ps1. It also lists other files in the workspace: .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 $conn
13 # ----- Attributes for connection "2"
14 $connID = 2
```

- Next, we can show the creation of a new Server Profile by changing a few settings in the code.
- First, we need to connect to the appliance, add the following lines 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
```

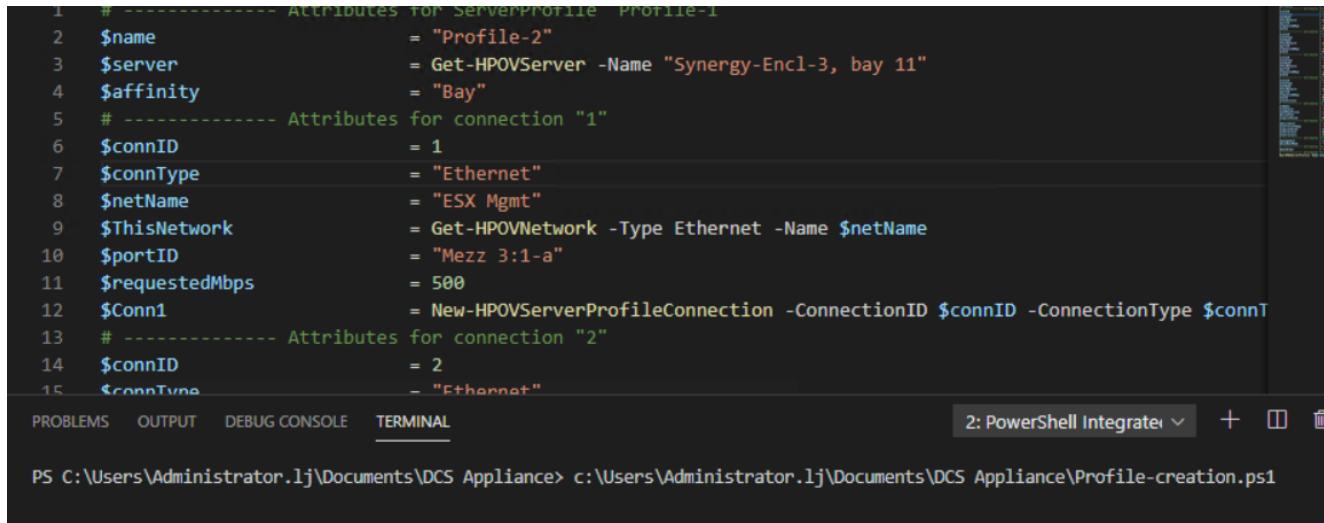
- Then change the following values:
  - \$name with **Profile-2**
  - \$server with **Synergy-Encl-2, bay 7**
  - The requested bandwidth of connection **1** and **2** to **7500** Mbps

```
PowerShell > > Profile-2-creation.ps1 > ...
11  Connect-HPOVMgmt -appliance $IP -Credential $credentials
12
13
14  # ----- Attributes for ServerProfile "Profile-2"
15  $name          = "Profile-2"
16  $description   = "Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL"
17  $server         = Get-HPOVServer -Name "Synergy-Encl-2, bay 7"
18  $affinity       = "Bay"
19  # ----- Attributes for connection "1"
20  $connID        = 1
21  $connName      = "Prod-NetworkSet-1"
22  $connType       = "Ethernet"
23  $netName        = "Prod"
24  $ThisNetwork    = Get-HPOVNetworkSet -Name $netName
25  $portID         = "Mezz 3:1-c"
26  $requestedMbps  = 7500
27  $Conn1          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connTy
28  # ----- Attributes for connection "2"
29  $connID        = 2
30  $connName      = "Prod-NetworkSet-2"
31  $connType       = "Ethernet"
32  $netName        = "Prod"
33  $ThisNetwork    = Get-HPOVNetworkSet -Name $netName
34  $portID         = "Mezz 3:2-c"
35  $requestedMbps  = 7500
36  $Conn2          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connTy
37  # ----- Attributes for connection "3"
38  $connID        = 3
39  $connName      = "Deployment Network A"
40  $connType       = "Ethernet"
41  $netName        = "Deployment"
42  $ThisNetwork    = Get-HPOVNetwork -Type Ethernet -Name $netName
43  $portID         = "Mezz 3:1-a"
44  $requestedMbps  = 2500
45  $bootPriority   = "Primary"
46  $volSource      =
47  $Conn3          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connTy
48  # ----- Attributes for connection "4"
49  $connID        = 4
50  $connName      = "Deployment Network B"
51  $connType       = "Ethernet"
52  $netName        = "Deployment"
53  $ThisNetwork    = Get-HPOVNetwork -Type Ethernet -Name $netName
54  $portID         = "Mezz 3:2-a"
55  $requestedMbps  = 2500
56  $bootPriority   = "Secondary"
57  $volSource      =
58  $Conn4          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionType $connTy
59  $connections    = $Conn1, $Conn2, $Conn3, $Conn4
60  # ----- Attributes for logical disk "SAS RAID1 SSD(RAID1)"
```

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

```
Disconnect-HPOVMgmt
```

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



```

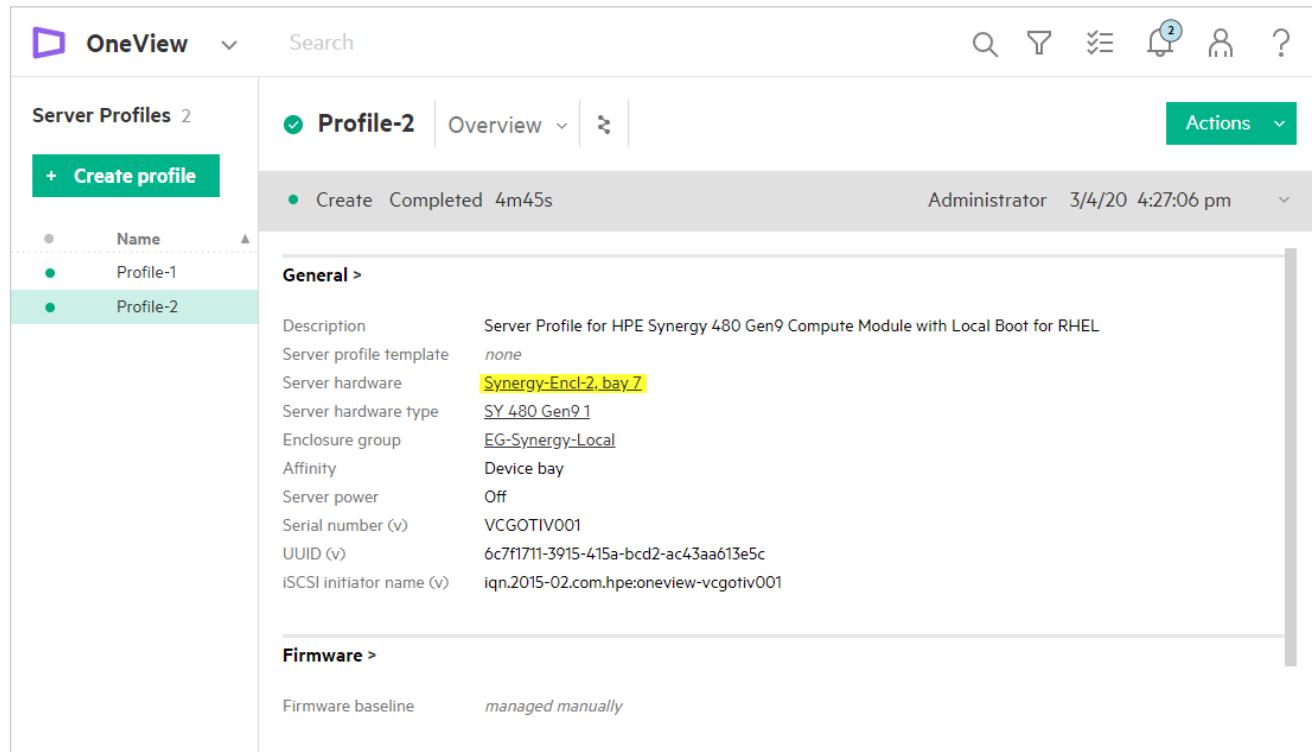
1 # ----- Attributes for ServerProfile: Profile-1
2 $name = "Profile-2"
3 $server = Get-HPOVServer -Name "Synergy-Encl-3, bay 11"
4 $affinity = "Bay"
5 # ----- Attributes for connection "1"
6 $connID = 1
7 $connType = "Ethernet"
8 $netName = "ESX Mgmt"
9 $ThisNetwork = Get-HPOVNetwork -Type Ethernet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

```

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



General >	
Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	<i>none</i>
Server hardware	Synergy-Encl-2, bay 7
Server hardware type	SY 480 Gen9 1
Enclosure group	EG-Synergy-Local
Affinity	Device bay
Server power	Off
Serial number (v)	VCGOTIV001
UUID (v)	6c7f1711-3915-415a-bcd2-ac43aa613e5c
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgotiv001

As illustrated above, the profile has been assigned to **Synergy-Encl-2, bay 7** as requested in our code



# Hewlett Packard Enterprise

- 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, the 'Server Profiles' list shows 'Profile-1' and 'Profile-2'. 'Profile-2' is selected and highlighted with a red box and number 1. In the center, the 'Connections' tab is selected under 'Profile-2' (highlighted with a red box and number 2). The 'Connections' table lists several network sets. Connection 1, 'Prod-NetworkSet-1', is expanded (indicated by a red circle with number 3). The 'Requested bandwidth' field for this connection is highlighted with a red box and number 4, showing the value '7.5 Gb/s'.

ID	Name	Network	Port	Boot
1	Prod-NetworkSet-1	Prod (network set)	Mezzanine 3:1-c	Not bootable
2	Prod-NetworkSet-2	Prod (network set)	Mezzanine 3:2-c	Not bootable
3	Deployment Network A	Deployment VLAN1500	Mezzanine 3:1-a	Not bootable
4	Deployment Network B	Deployment VLAN1500	Mezzanine 3:2-a	Not bootable

This concludes PowerShell – Scenario 1



# Hewlett Packard Enterprise

## 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

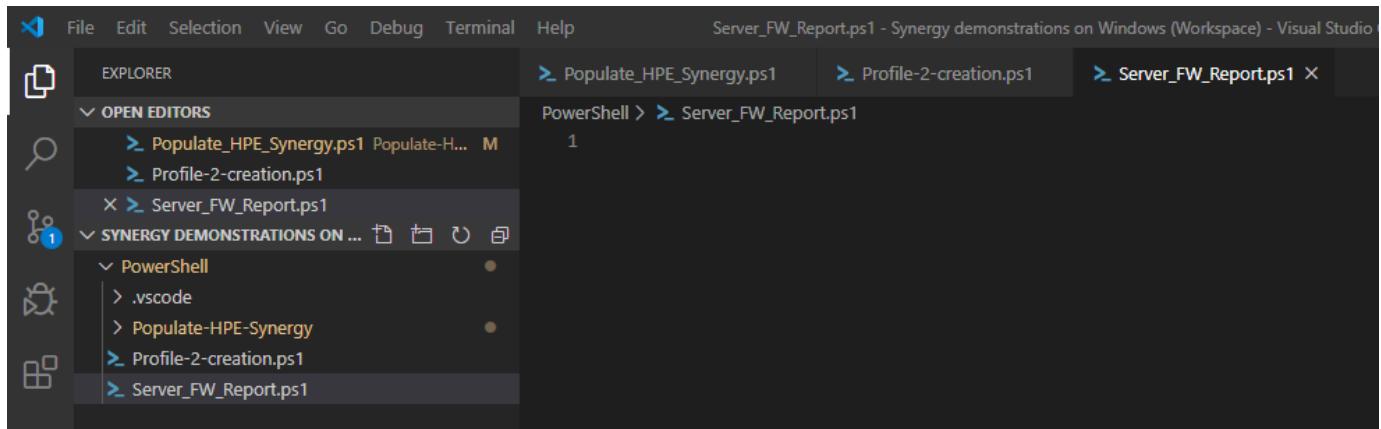
The screenshot shows the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar says "Profile-2-creation.ps1". The left sidebar has icons for Explorer, Open Editors, and Synergy Demonstrations on ... (with a count of 1). The "OPEN EDITORS" section shows two files: "Populate\_HPE\_Synergy.ps1" and "Profile-2-creation.ps1". A red box highlights the "New File" button in the "SYNTERGY DEMONSTRATIONS ON ..." section. The main editor area displays PowerShell code:

```
62 $deviceSlot
63 $controllerMode
64 $LogicalDisks
65 $controller1
66 $controllers
67 # ----- Attril
68 $manageboot
69 $biosBootMode
70 # ----- Attril
71 $bootOrder
72 # ----- Attril
73 New-HPOVServerProfile
74
75
76 Disconnect-HPOVMgmt
```

- Name it **Server\_FW\_Report.ps1**



# Hewlett Packard Enterprise



- 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
    }
}
```

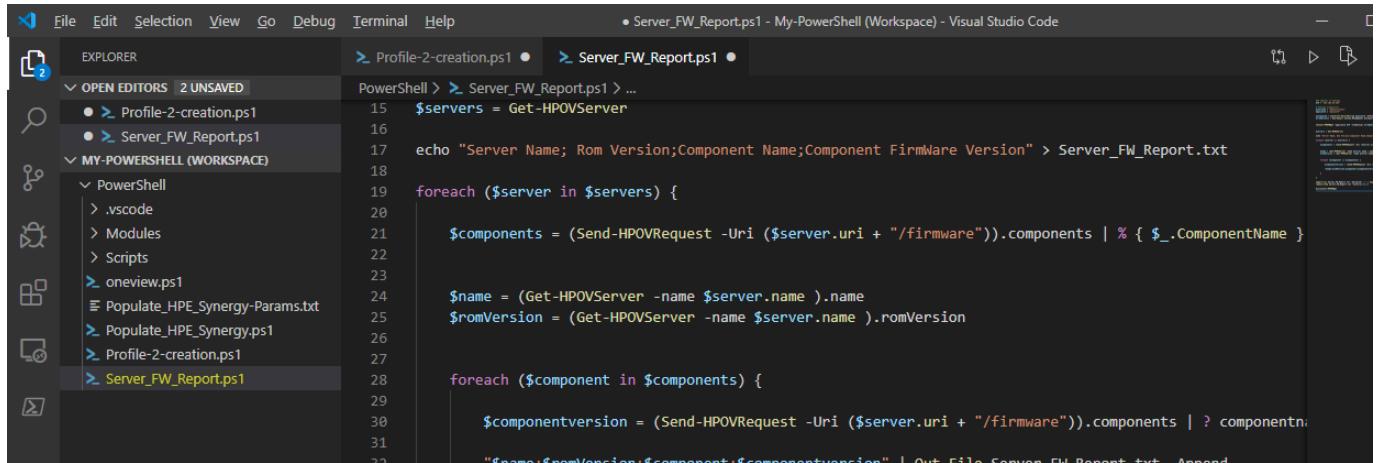


# Hewlett Packard Enterprise

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

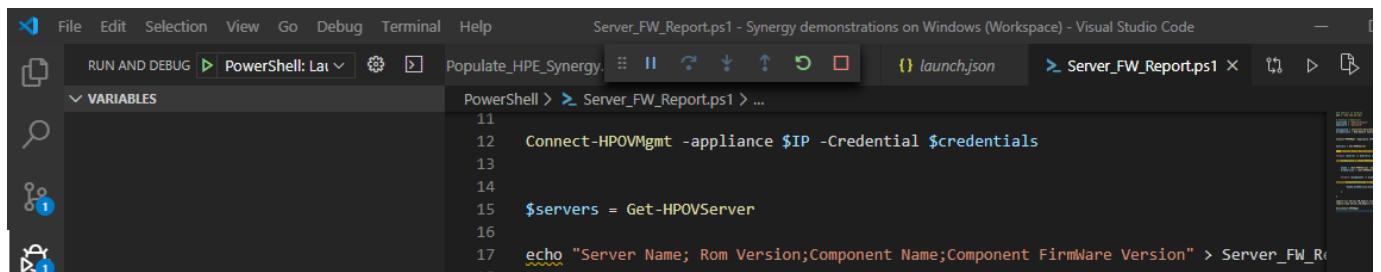
```
Disconnect-HPOVMgmt
```

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



```
File Edit Selection View Go Debug Terminal Help • Server_FW_Report.ps1 - My-PowerShell (Workspace) - Visual Studio Code  
EXPLORER OPEN EDITORS 2 UNSAVED PowerShell > Server_FW_Report.ps1 > ...  
Profile-2-creation.ps1  
Server_FW_Report.ps1  
MY-POWERSHELL (WORKSPACE)  
PowerShell  
.vscode  
Modules  
Scripts  
oneview.ps1  
Populate_HPE_Synergy-Params.txt  
Populate_HPE_Synergy.ps1  
Profile-2-creation.ps1  
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     $name = (Get-HPOVServer -name $server.name).name  
24     $romVersion = (Get-HPOVServer -name $server.name).romVersion  
25  
26     foreach ($component in $components) {  
27  
28         $componentversion = (Send-HPOVRequest -Uri ($server.uri + "/firmware")).components | ? componentn  
29         "$name:$romVersion:$component:$componentversion" | Out-File Server_FW_Report.txt -Append  
30  
31  
32
```

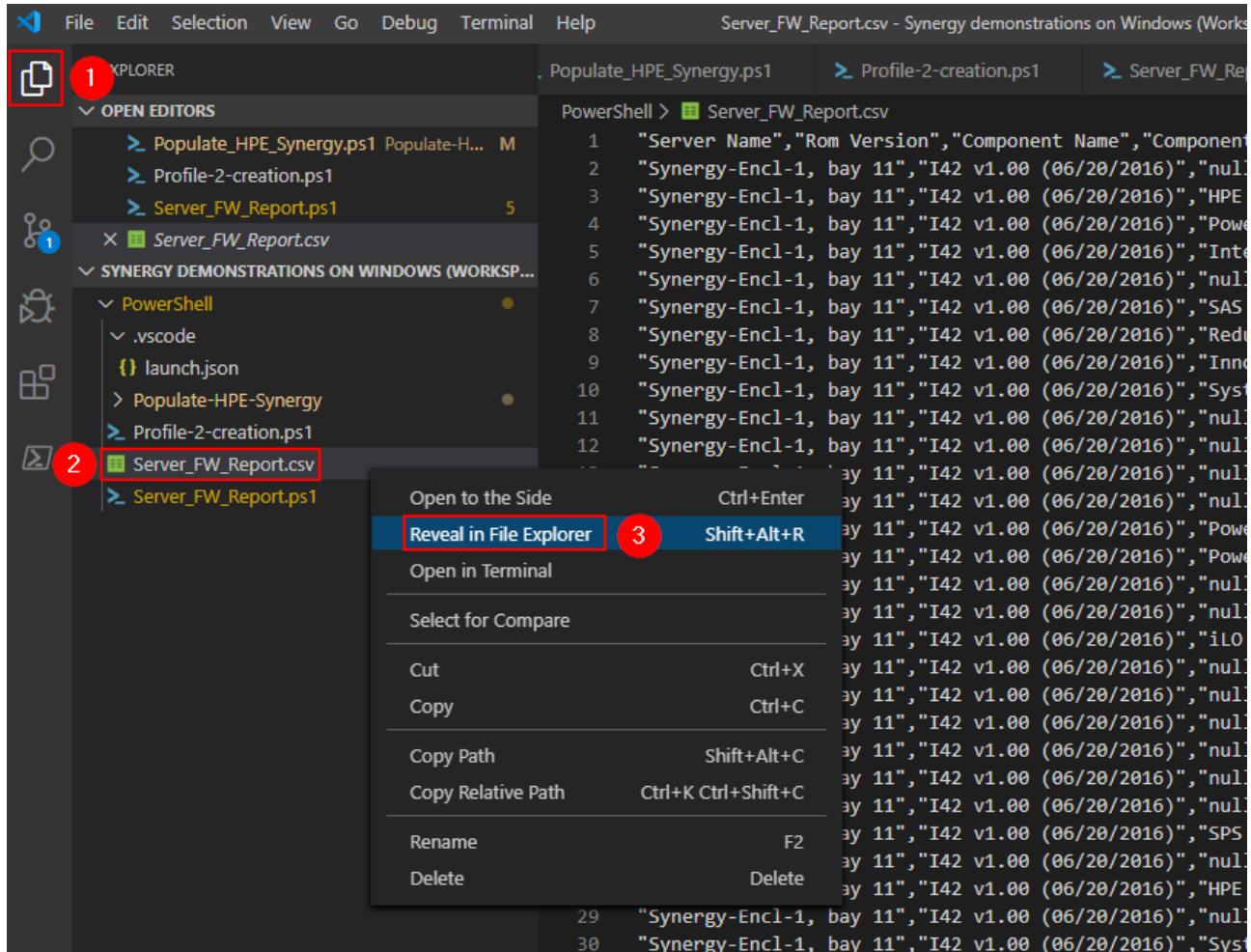
- Then we can run the script by pressing **F5**



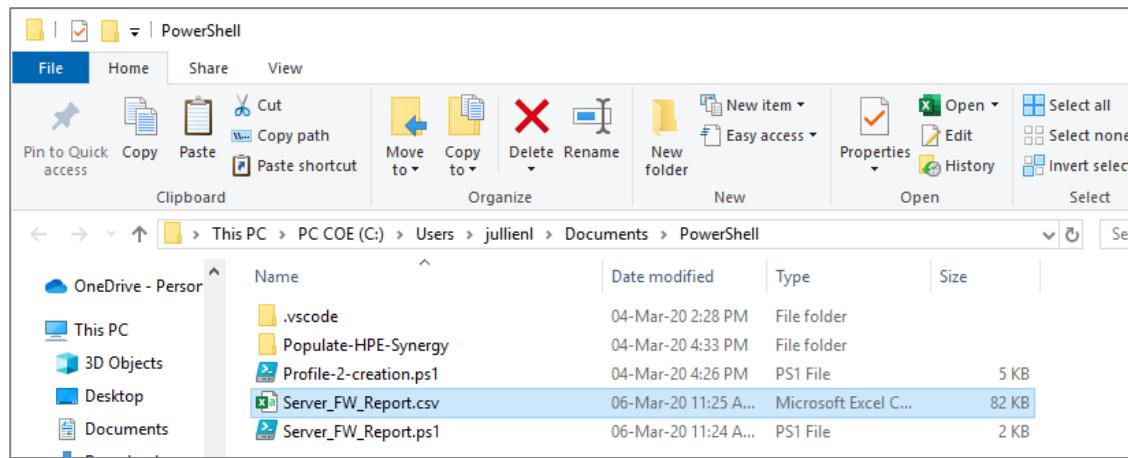
```
File Edit Selection View Go Debug Terminal Help Server_FW_Report.ps1 - Synergy demonstrations on Windows (Workspace) - Visual Studio Code  
RUN AND DEBUG PowerShell: Launched Variables PowerShell > Server_FW_Report.ps1 > ...  
Populate_HPE_Synergy.ps1 launch.json Server_FW_Report.ps1  
11  
12 Connect-HPOVMgmt -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





# Hewlett Packard Enterprise

- 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	Server Name
34	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null
35	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	ME SPI Descriptor
36	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null
37	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null
38	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Intelligent Provisioning
39	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HP ProLiant System ROM - Backup
40	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	System Programmable Logic Device
41	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Intelligent Platform Abstraction Data
42	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Power Management Controller Firmware
43	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Power Management Controller FW Bootloader
44	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HPE Synergy 3820C 10/20Gb Converged Network Adapter
45	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HPE Synergy 3530C 16Gb Fibre Channel Host Bus Adapter
46	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	Server Platform Services (SPS) Firmware
47	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HP ProLiant System ROM
48	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	iLO
49	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	System Programmable Logic Device
50	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Power Management Controller Firmware
51	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Power Management Controller FW Bootloader
52	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HPE Synergy 3820C 10/20Gb Converged Network Adapter
53	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HPE Synergy 3830C 16G Fibre Channel Host Bus Adapter
54	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Intelligent Provisioning
55	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HP ProLiant System ROM
56	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	iLO
57	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Intelligent Platform Abstraction Data
58	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	HP ProLiant System ROM - Backup
59	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Server Platform Services (SPS) Firmware
60	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Intelligent Provisioning
61	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	iLO
62	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	HPE Synergy 3820C 10/20Gb Converged Network Adapter
63	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	HP ProLiant System ROM - Backup
64	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Power Management Controller FW Bootloader
65	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	HP ProLiant System ROM
66	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	System Programmable Logic Device
67	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Server Platform Services (SPS) Firmware
68	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Intelligent Platform Abstraction Data
69	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	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 a terminal window titled 'PowerShell Integrated' with the following command history:

```
PS C:\Users\jullienl\Documents\PowerShell> Add-Content networks_creation.csv -value '"NetName","VLAN_ID"'

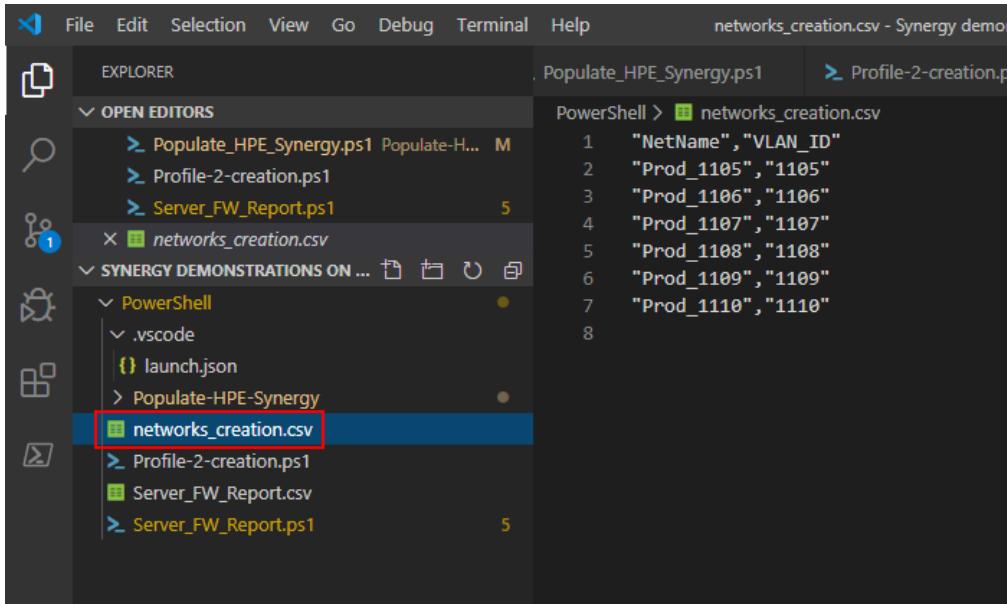
PS C:\Users\jullienl\Documents\PowerShell> $networks = @("Prod_1105","1105","","Prod_1106","1106","","Prod_1107","1107","","Prod_1108","1108","","Prod_1109","1109",
  ,,"Prod_1110","1110"    )

PS C:\Users\jullienl\Documents\PowerShell> $networks | foreach { Add-Content -Path  networks_creation.csv -Value $_ }
```

- This script creates a **networks\_creation.csv** in the working directory



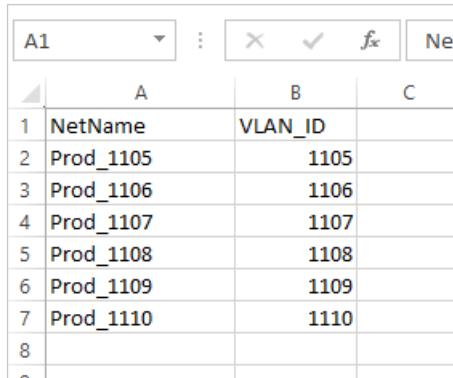
# Hewlett Packard Enterprise



The screenshot shows the Visual Studio Code interface. The left sidebar is titled 'EXPLORER' and lists several files under 'OPEN EDITORS' and 'SYNTERGY DEMONSTRATIONS ON ...'. The file 'networks\_creation.csv' is highlighted with a red box. The right pane shows the contents of the CSV file:

```
PowerShell > networks_creation.csv
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
```

In Excel, it is built as follows:



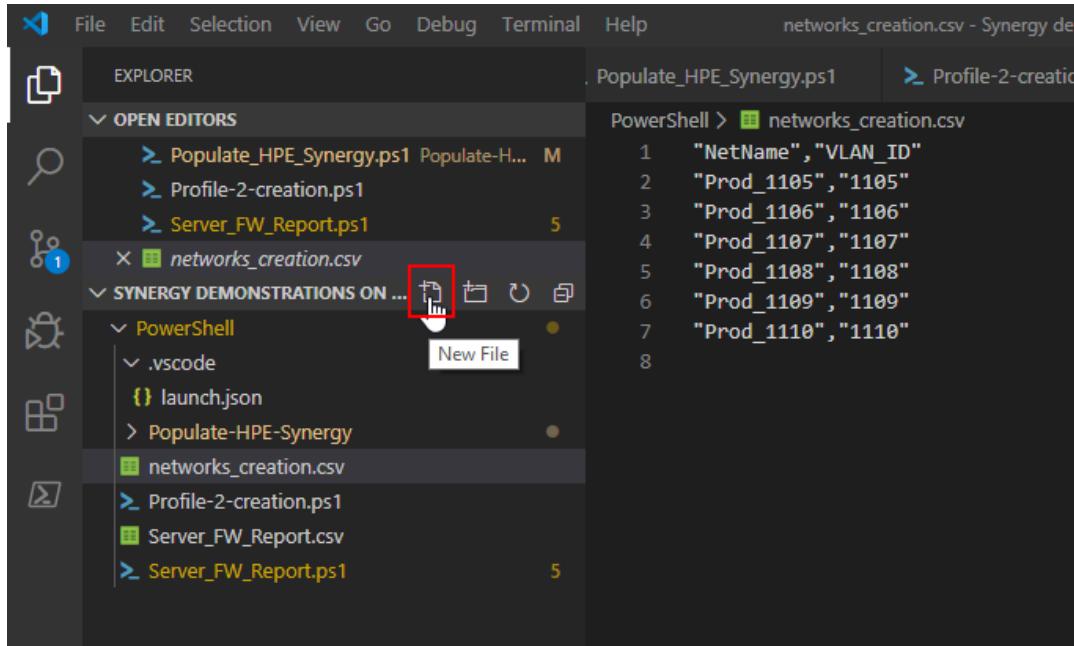
	A	B	C
1	NetName	VLAN_ID	
2	Prod_1105	1105	
3	Prod_1106	1106	
4	Prod_1107	1107	
5	Prod_1108	1108	
6	Prod_1109	1109	
7	Prod_1110	1110	
8			

This is the spreadsheet that Power Shell will be used to generate six Ethernet Networks **Prod\_1105**, **Prod\_1106**,... **Prod\_1110** In our Synergy environment.



# Hewlett Packard Enterprise

- 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 = "US-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:

```
Creating Network: Prod_1105
Adding Network: Prod_1105 to Uplink Set: US-Prod

Creating Network: Prod_1106
Adding Network: Prod_1106 to Uplink Set: US-Prod

Creating Network: Prod_1107
Adding Network: Prod_1107 to Uplink Set: US-Prod

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

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

Creating Network: Prod_1110
Adding Network: Prod_1110 to Uplink Set: US-Prod
```



# Hewlett Packard Enterprise

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

The screenshot shows the HPE OneView interface with the 'Networks' page open. On the left, a table lists 21 networks, with the first one, 'Deployment', selected and highlighted in green. A green button labeled '+ Create network' is visible. On the right, the 'Deployment' network's details are displayed in a card, showing its configuration under the 'General' tab. The 'Type' is listed as 'Ethernet', 'VLAN' as '1500', and 'Associated with IPv4 subnet ID' as '10.11.0'. Other fields include 'Associated with IPv6 subnet ID' (set to 'none'), 'Purpose' (set to 'General'), 'Preferred bandwidth' (2.5 Gb/s), 'Maximum bandwidth' (20 Gb/s), 'Smart link' (set to 'Yes'), 'Private network' (set to 'No'), 'Uplink set' (set to 'US-Image Streamer'), and 'Used by' (set to '1 deployment cluster'). The 'Member of' field indicates 'no network sets'.

Name	VLAN	Type
Deployment	1500	Ethernet
ESX Mgmt	1131	Ethernet
ESX vMotion	1132	Ethernet
Mgmt	100	Ethernet
Prod_1101	1101	Ethernet
Prod_1102	1102	Ethernet
Prod_1103	1103	Ethernet
Prod_1104	1104	Ethernet
Prod_1105	1105	Ethernet
Prod_1106	1106	Ethernet
Prod_1107	1107	Ethernet
Prod_1108	1108	Ethernet
Prod_1109	1109	Ethernet
Prod_1110	1110	Ethernet
SAN A FC		FC
SAN A FCoE	10	FCoE
SAN B FC		FC
SAN B FCoE	11	FCoE



# Hewlett Packard Enterprise

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

The screenshot shows the HPE OneView interface. On the left, under 'Logical Interconnect Groups' (1), there is a list with 'LIG-FC' (2) selected. On the right, under 'Uplink Sets' (3), the 'US-Prod' set is expanded (4). It shows configuration details like Consistency checking (Exact match), Connection mode (Automatic), LACP timer (Short (1s)), LACP load balancing (Source & Destination MAC Address), and Native network (none). Below this, a list of 'Networks (10)' is shown, with Prod\_1101 through Prod\_1110 listed.

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

```
Update from group LF-Synergy-Local-LIG-FlexFabric
Claim interconnect
[oooooooooooooooooooooooooooo]

Creating Network: Prod_1106
Adding Network: Prod_1106 to Uplink Set: US-Prod

Creating Network: Prod_1107
Adding Network: Prod_1107 to Uplink Set: US-Prod

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

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

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

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

Please wait...

```

Ln 95, Col 3 Spaces: 4 UTF-8 CRLF PowerShell □ 5.1 ⌂ ⌂



# Hewlett Packard Enterprise

- 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):

▼ ▲ *The logical interconnect is inconsistent with the logical interconnect group LIG-FlexFabric.*

1/29/20 8:58:34 Cleared unassigned ▾  
am  
20 minutes ago

**Resolution** The logical interconnect can be made consistent with its logical interconnect group via update from group. The logical interconnect or group may also be edited to manually restore consistency. If consistency is not desired, this alert can be cleared.

Notes

- After a few minutes, the LI turns Green

OneView Search

Logical Interconnects 7

LE-Synergy-Local-LIG-FlexFabric Logical Interconnect

Update from group  Configure interconnect

**Logical Interconnect**

<i>Internal</i> no networks	US-SAN-... 1 network 1 uplink port	US-ESX-... 1 network 2 uplink ports	US-ESX-v... 1 network 2 uplink ports
--------------------------------	--	---	--

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

The screenshot shows the HPE OneView interface. In the top navigation bar, 'Logical Interconnects' is selected (highlighted with a red box and number 1). The main content area displays a list of logical interconnects under the heading 'Uplink Sets'. A specific entry, 'LE-Synergy-Local-LIG-FlexFabric', is highlighted with a red box and number 2. Below it, an 'Uplink Sets' section is expanded, showing configuration details like Connection mode (Automatic), LACP timer (Short (1s)), and LACP load balancing (Source & Destination MAC Address). The 'US-Prod' network set is selected (highlighted with a red box and number 4). The 'Networks (10)' section lists ten network VLAN IDs: Prod\_1101 through Prod\_1110. The last two entries, Prod\_1109 and Prod\_1110, are highlighted with yellow boxes.

- 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

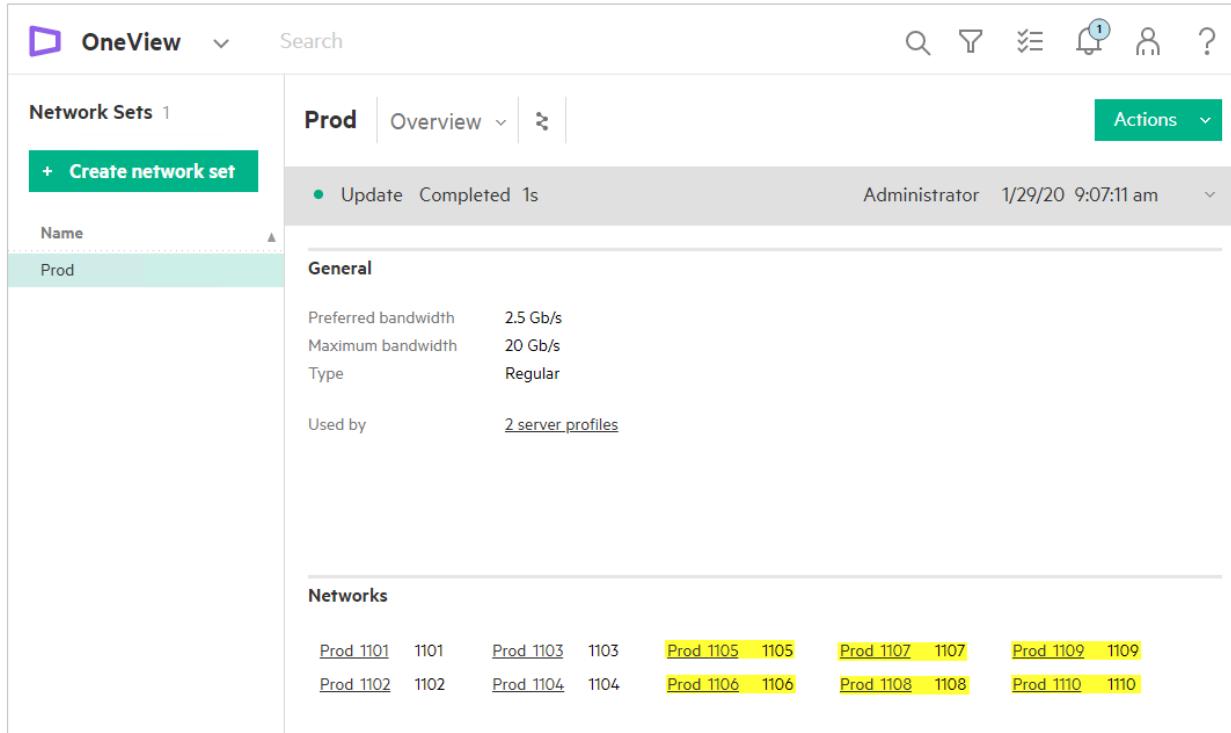
Adding Network: Prod_1108 to NetworkSet: Prod
The network VLAN ID: Prod_1108 has been added successfully to all Server Profiles that are using the Network Set: Prod

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>

```

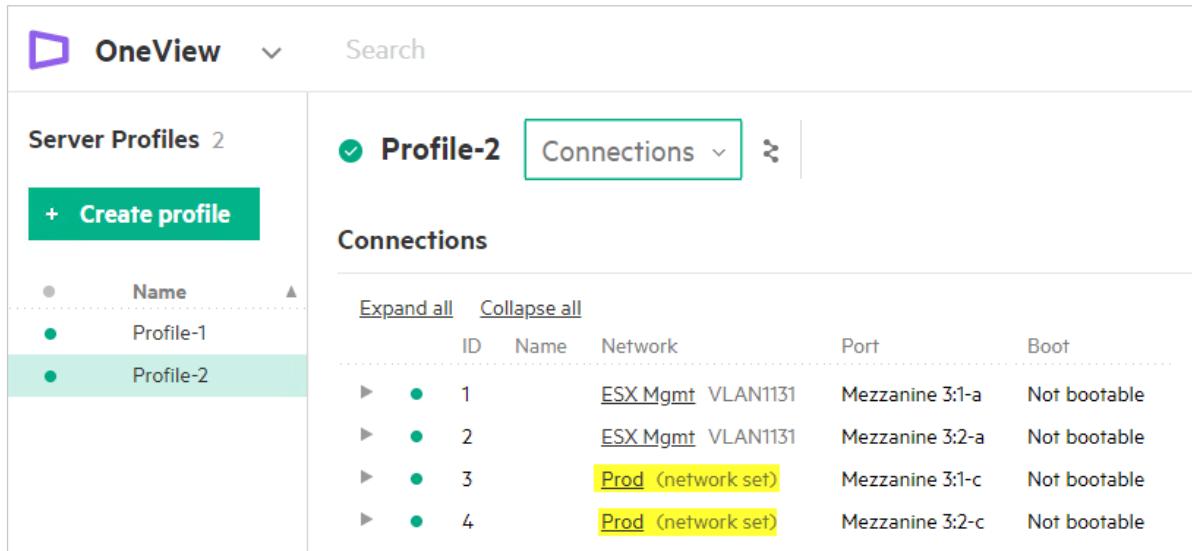
- You can show in OneView, the additional networks



The screenshot shows the OneView interface for managing network sets. On the left, a sidebar lists 'Network Sets 1' with a '+ Create network set' button. The main panel is titled 'Prod' under 'Overview'. It displays a summary message: 'Update Completed 1s' by 'Administrator' on '1/29/20 9:07:11 am'. Below this, the 'General' section shows details: Preferred bandwidth (2.5 Gb/s), Maximum bandwidth (20 Gb/s), Type (Regular), and 'Used by' (2 server profiles). The 'Networks' section lists several network connections, with the 'Prod' network set highlighted in yellow.

ID	Name	Network	Port	Boot
1	ESX Mgmt	VLAN1131	Mezzanine 3:1-a	Not bootable
2	ESX Mgmt	VLAN1131	Mezzanine 3:2-a	Not bootable
3	Prod (network set)		Mezzanine 3:1-c	Not bootable
4	Prod (network set)		Mezzanine 3:2-c	Not bootable

- 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 OneView interface for managing server profiles. On the left, a sidebar lists 'Server Profiles 2' with a '+ Create profile' button. The main panel is titled 'Profile-2' under 'Connections'. It shows a 'Connections' table with four entries. The last two entries, IDs 3 and 4, are connected to the 'Prod' network set, which is highlighted in yellow.

ID	Name	Network	Port	Boot
1	ESX Mgmt	VLAN1131	Mezzanine 3:1-a	Not bootable
2	ESX Mgmt	VLAN1131	Mezzanine 3:2-a	Not bootable
3	Prod (network set)		Mezzanine 3:1-c	Not bootable
4	Prod (network set)		Mezzanine 3:2-c	Not bootable

This concludes PowerShell – Scenario 3



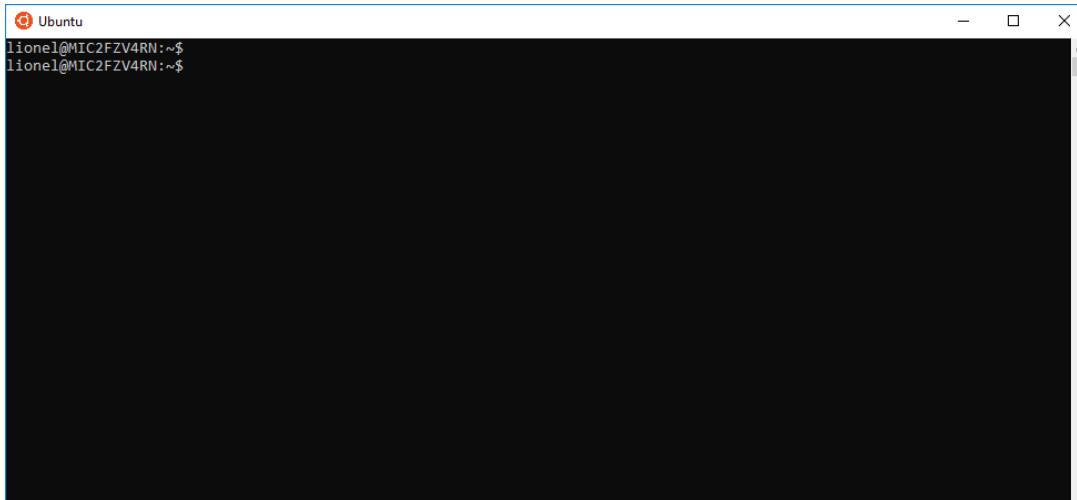
# Hewlett Packard Enterprise

## 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:

- Make sure your Ubuntu session running in WSL is started.



- 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)



# Hewlett Packard Enterprise

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:

The screenshot shows the Visual Studio Code interface. The left sidebar (Explorer) displays the project structure. A red box labeled 1 highlights the 'OPEN EDITORS' section, which contains 'Profile-3-creation.py' and 'enclosures.py'. Another red box labeled 2 highlights the 'oneview-python' folder under 'MY-WSL-UBUNTU (WORKSPACE)'. A third red box labeled 3 highlights the 'examples' folder. A fourth red box labeled 4 highlights the 'enclosures.py' file within the examples folder. The right pane shows the code content of 'enclosures.py'.

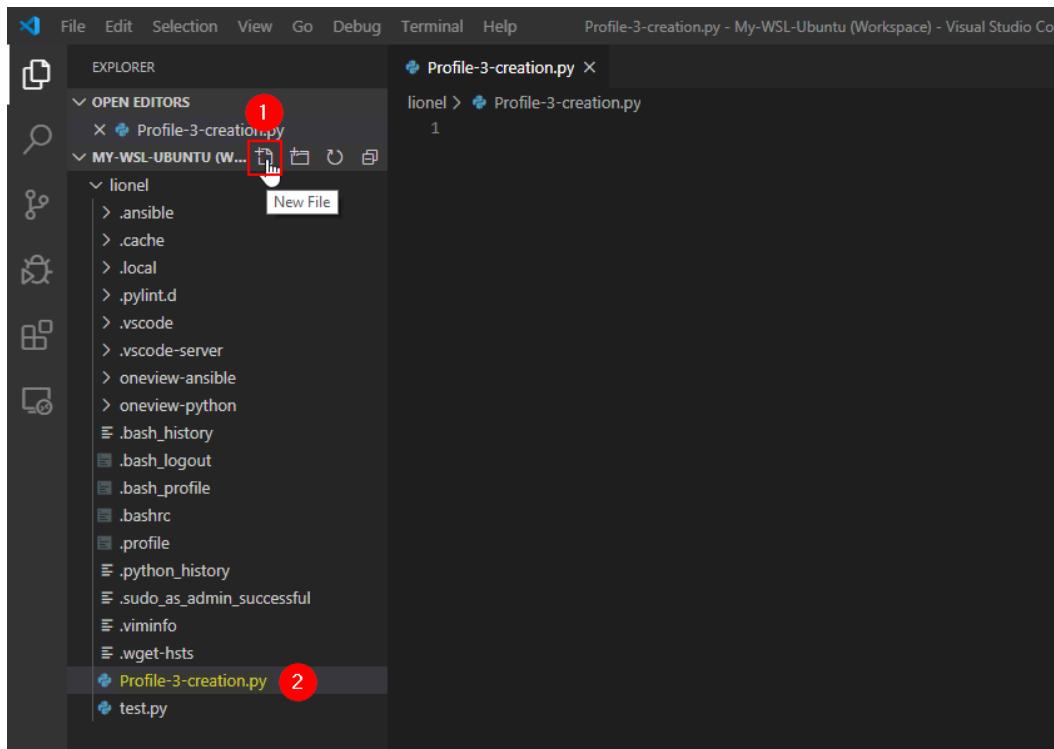
```
Profile-3-creation.py • enclosures.py X
lionel > oneview-python > examples > enclosures.py > ...
1  # -*- coding: utf-8 -*-
2  ###
3  # (C) Copyright [2019] Hewlett Packard Enterprise Development LP
4  #
5  # Licensed under the Apache License, Version 2.0 (the "License");
6  # you may not use this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

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



# Hewlett Packard Enterprise

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





# Hewlett Packard Enterprise

- 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 7')

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)
```



# Hewlett Packard Enterprise

- Turn the server on:

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

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



```
Profile-3-creation.py ●  
lionel > Profile-3-creation.py > ...  
1  from hpOneView.oneview_client import OneViewClient  
2  from pprint import pprint  
3  
4  config = {  
5      "ip": "192.168.56.101",  
6      "api_version": 1200,  
7      "credentials": {  
8          "userName": "Administrator",  
9          "password": "password"  
10     }  
11 }  
12  
13 oneview_client = OneViewClient(config)  
14  
15 server_hardwares = oneview_client.server_hardware  
16  
17 server_profile_templates = oneview_client.server_profile_templates  
18  
19 myspt = server_profile_templates.get_by_name(  
20     'HPE Synergy 480 Gen9 with Local Boot for RHEL Template')  
21  
22 server = server_hardwares.get_by_name('Synergy-Encl-3, bay 7')  
23  
24 profile = myspt.get_new_profile()  
25  
26 profile['serverHardwareUri'] = server.data['uri']  
27  
28 profile['name'] = 'Profile-3'  
29  
30 oneview_client.server_profiles.create(profile)  
31  
32 server_profiles = oneview_client.server_profiles
```



# Hewlett Packard Enterprise

- 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 (Python: Current), VARIABLES, and several other icons.
- Code Editor:** The file `Profile-3-creation.py` is open, displaying Python code for initializing an OneViewClient. The code defines a `config` dictionary with `ip`, `api_version`, and `credentials` (username and password). It then creates an `oneview_client` and initializes `server_hardwares` and `server_profile_templates`.
- Output Bar:** OUTPUT, TERMINAL, DEBUG CONSOLE, PROBLEMS. The TERMINAL tab shows command-line output related to the Python environment and the execution of the script.

```
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
```

```
/usr/bin/python3 /home/lione1/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lione1/Profile-3-creation.py
lione1@MIC2ZV4RN:~$ /usr/bin/python3 /home/lione1/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new_ptvsd/no_wheels/ptvsd/launcher /home/lione1/Profile-3-creation.py
```

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

OneView

Server Profiles 2

+ Create profile

Name

Profile-3

Profile-3

Overview

Create

Apply settings to Synergy-Encl-3, bay 7.

General

Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	<a href="#">HPE Synergy 480 Gen9 with Local Boot for RHEL Template</a>
Server hardware	<a href="#">Synergy-Encl-3, bay 7</a>
Server hardware type	<a href="#">SY 480 Gen9 1</a>
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	Device bay
Server power	Off
Serial number (v)	VCGOTIV008
UUID (v)	d86c73/a-/07c-/7d2-b001-0/59af15/8/





# Hewlett Packard Enterprise

- Once created, the server is turned on:

The screenshot shows the HPE OneView interface. In the top left, it says "OneView". The top right has a search bar and various navigation icons. On the left, there's a sidebar titled "Server Profiles 1" with a green button "+ Create profile". The main area shows a profile named "Profile-3" with a green checkmark. It says "Create Completed 4m34s" and was done by "Administrator" on "2/26/20 11:10:54 am". Below this, under "General > Edit", are the following details:

Description	Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL
Server profile template	<a href="#">HPE Synergy 480 Gen9 with Local Boot for RHEL Template</a>
Server hardware	Synergy-Encl-3,bay.7
Server hardware type	SY 480 Gen9 1
Enclosure group	<a href="#">EG-Synergy-Local</a>
Affinity	Device bay
Server power	On
Serial number (v)	VCGOTIV000
UUID (v)	9d2791e4-b3df-4bae-ab3b-0a87ccff8d9

This concludes Python – Scenario 1



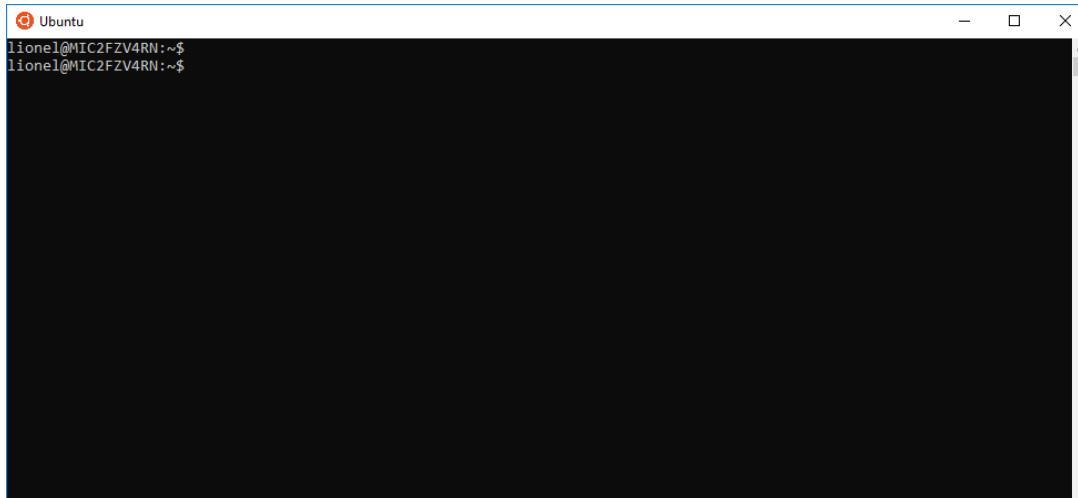
## Python – Scenario 2 – Creating a report

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:

- Make sure your Ubuntu session running in WSL is started.



- 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:

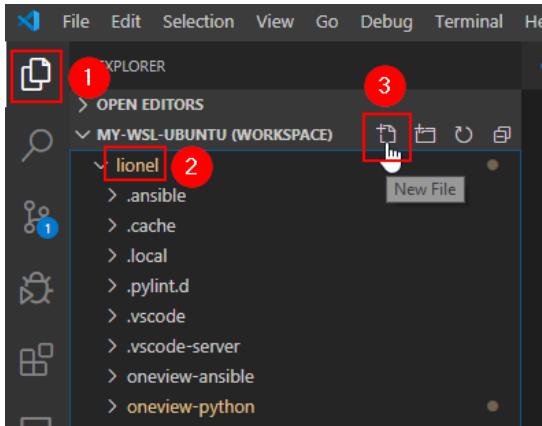


- **Notice:** This scenario can be run with either the initial or final configuration snapshot.



# Hewlett Packard Enterprise

- 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")

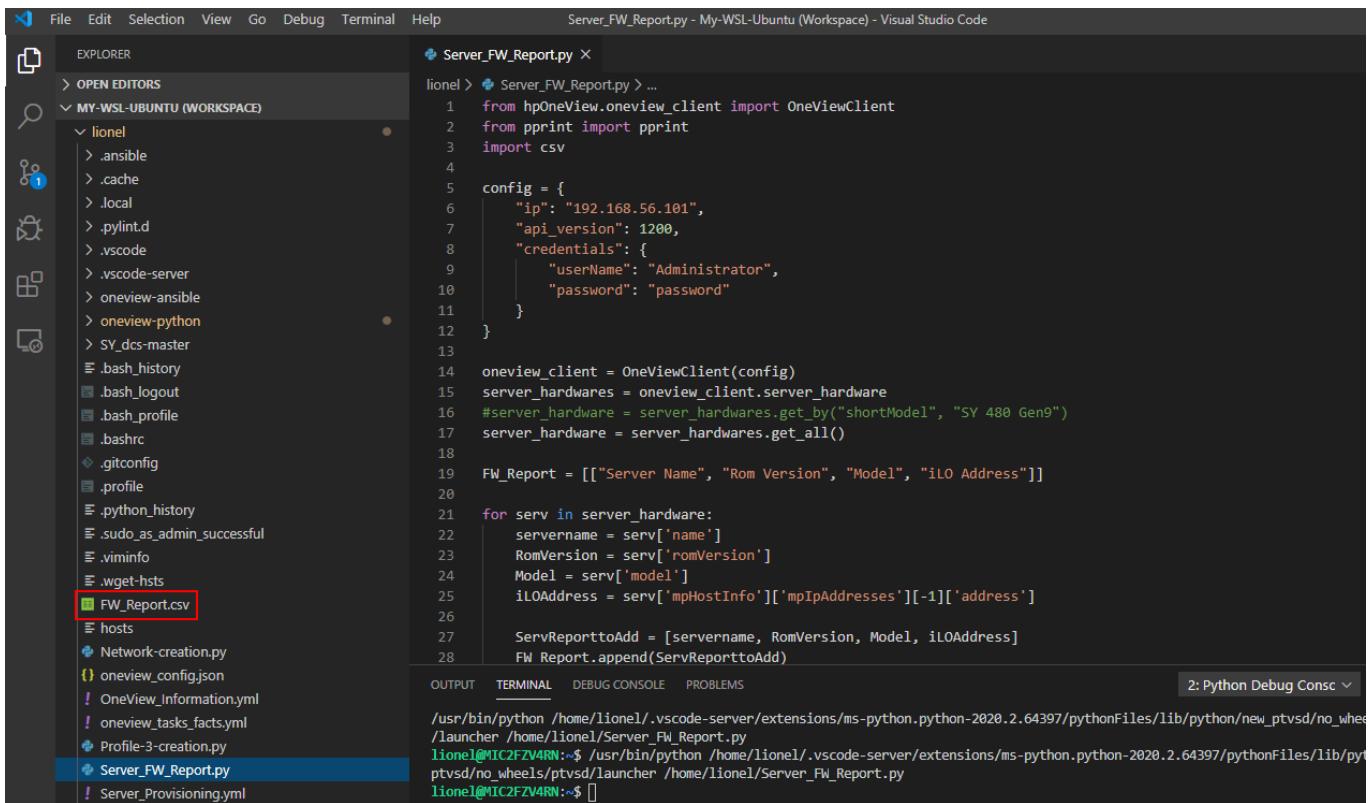
FW_Report = [["Server Name", "Rom Version", "Model", "iLO Address"]]

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', 'wb') as file:
    writer = csv.writer(file)
    writer.writerows(FW_Report)
```

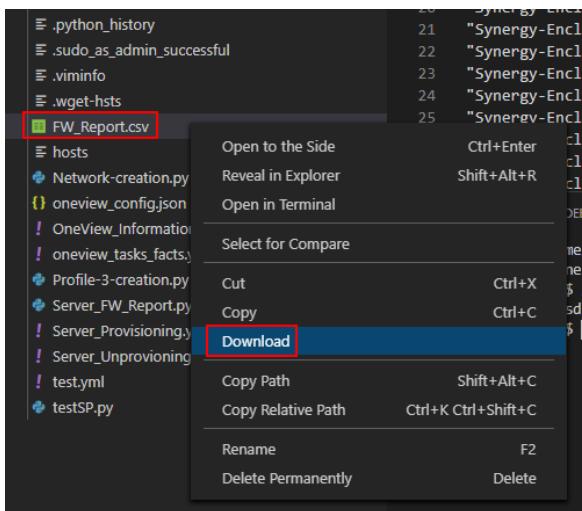
- Then save the script by pressing **CTRL + S**, then **F5** to run the script.
- Once run, you should find the **FW\_Report.csv** file in your user home directory:



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

- File Explorer (Left):** Shows the workspace structure under "MY-WSL-UBUNTU (WORKSPACE)". Files listed include .bash\_history, .bash\_logout, .bash\_profile, .bashrc, .gitconfig, .profile, .python\_history, .sudo\_as\_admin\_successful, .viminfo, .wget-hsts, FW\_Report.csv (highlighted with a red box), hosts, Network-creation.py, onerview\_config.json, OneView\_Information.yml, onerview\_tasks\_facts.yml, Profile-3-creation.py, Server\_FW\_Report.py (highlighted with a blue box), and Server\_Provisioning.yml.
- Code Editor (Right):** Displays the Python script "Server\_FW\_Report.py". The code uses the hpOneView library to interact with OneViewClient, retrieves server hardware information, and generates a CSV report. The output of the script execution is shown in the terminal below.
- Terminal (Bottom):** Shows the command "/usr/bin/python /home/lionel/.vscode-server/extensions/ms-python.python-2020.2.64397/pythonFiles/lib/python/new\_ptvsd/no\_wheels/launcher /home/lionel/Server\_FW\_Report.py" being run, followed by the output of the script execution.

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



# Hewlett Packard Enterprise

- 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

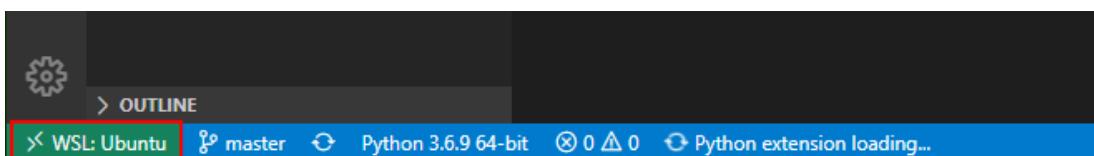
In this scenario, we are going to use Python to create a new network in OneView (Name: *RHEL Prod*, VLANID: 50), then add this network to the uplink set *US-Prod* and to the network set *Prod* so that any of the existing Server Profiles using the network set *Prod* will be automatically connected to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

### Prerequisites:

- Make sure your Ubuntu session running in WSL is started.

```
Ubuntu
lionel@MIC2FZV4RN:~$
```

- 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:

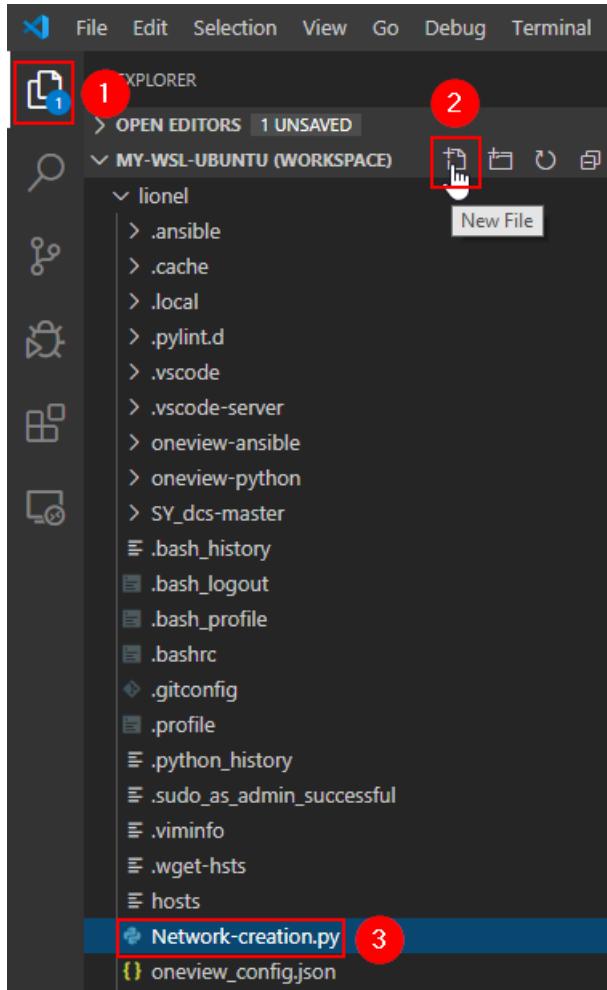


- **Import notice:** This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)



# Hewlett Packard Enterprise

- In VS Code, go to the **Explorer** to create in your user home directory a new file, name it **Network-creation.py**





# Hewlett Packard Enterprise

- Then add the following lines to your script:

```
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)

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 successfully.\n  uri = '%s' " %
      (ethernet_network.data['uri']))

# Adding network to Logical Interconnect Uplink Set

uplink_sets = oneview_client.uplink_sets

logical_interconnect_uri = oneview_client.logical_interconnects.get_all()[0]['uri']

ethernet_network_uri = ethernet_network.data['uri']

ethernet_network_name = options_ethernet['name']

uplink_set = uplink_sets.get_by_name("US-Prod")

uplink_added_ethernet = uplink_set.add_ethernet_networks(
    ethernet_network_name)

print("The uplink set with name = '{name}' have now the networks:\n {networkUris}".format(
    **uplink_added_ethernet))

# Adding new network to Network Set
```

```
network_set = network_sets.get_by_name("Prod")

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" %
      (network_set.data['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:

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

**RHEL Prod**

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



# Hewlett Packard Enterprise

- 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 HPE OneView interface for the Logical Interconnects page. A logical interconnect named "LE-Synergy-Local-LIG-FlexFabric" is selected. In the Uplink Sets section, the "US-Prod" set is expanded, showing its configuration and associated networks. The "RHEL Prod" network is listed under the "Networks" section.

Logical Interconnects 7 (1)

LE-Synergy-Local-LIG-FlexFabric (2)

Uplink Sets (3)

US-Prod (4)

RHEL Prod (highlighted in yellow)

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

The screenshot shows the HPE OneView interface for the Network Set page. A network set named "Prod" is selected. In the Networks section, the "RHEL Prod" network is listed.

Network Sets 1 (1)

+ Create network set

Prod (2)

Overview (3)

Administrator 2/26/20 3:08:25 pm

General

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

Networks

RHEL Prod (highlighted in yellow)



# Hewlett Packard Enterprise

- In the console, the following is displayed:

```
lionel@MIC2FZV4RN:~$ env PTVSD_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



# Hewlett Packard Enterprise

## 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.

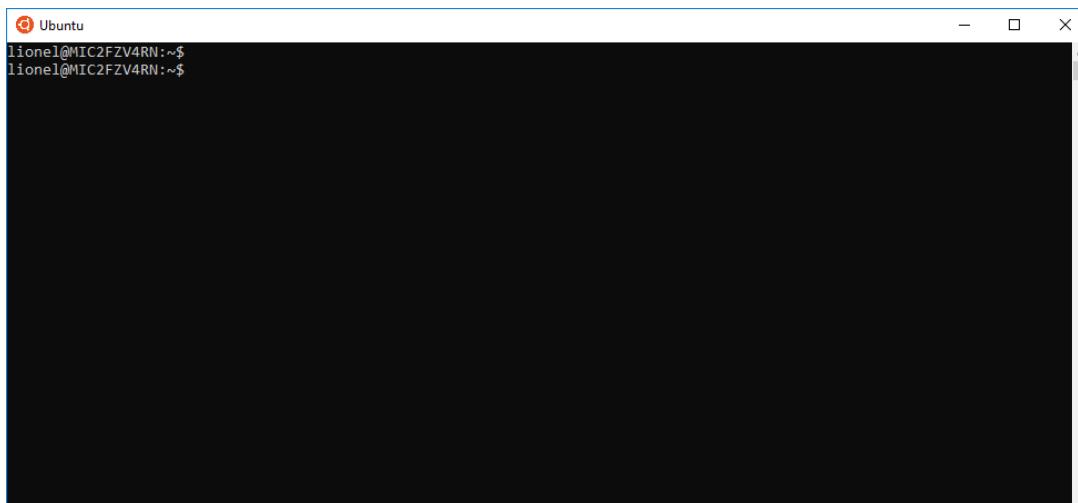
The integration of Ansible with HPE OneView extends the ability to automate the provisioning of bare-metal resources, including servers, storage, and networking. This accelerates time to value through automated, consistent provisioning.

**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:

- Make sure your Ubuntu session running in WSL is started.



- 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:

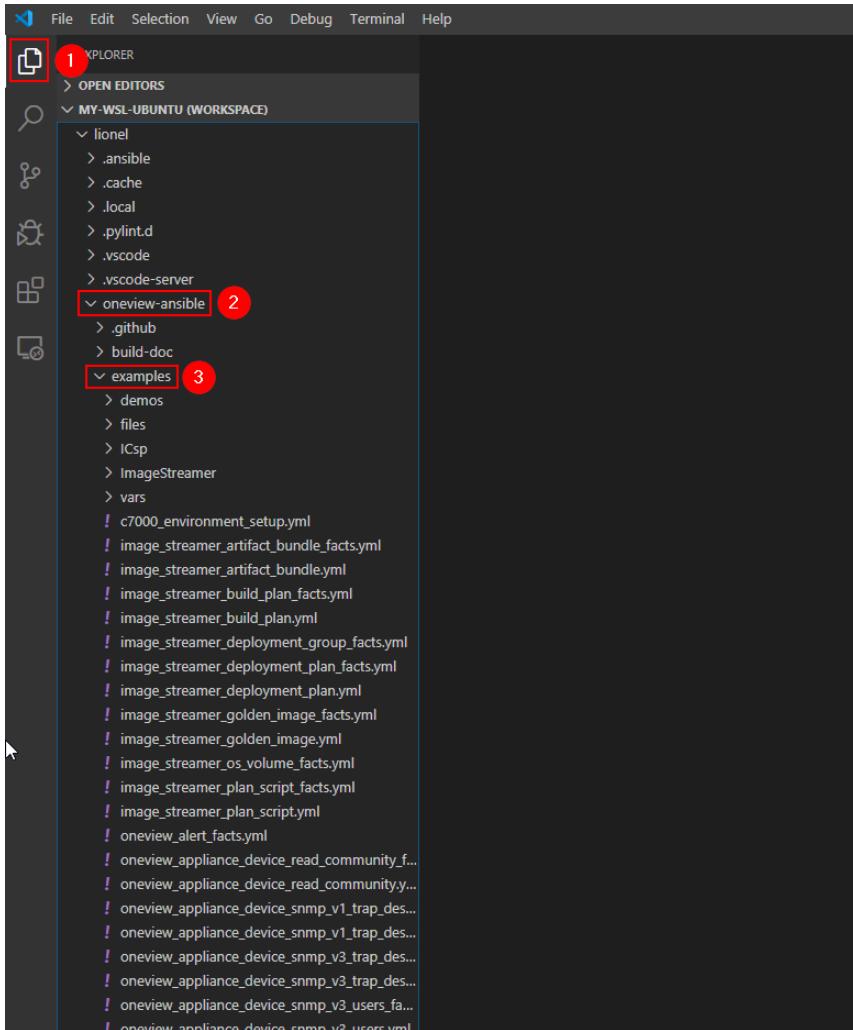


- **Notice:** This scenario can be run with either the initial or final configuration snapshot.



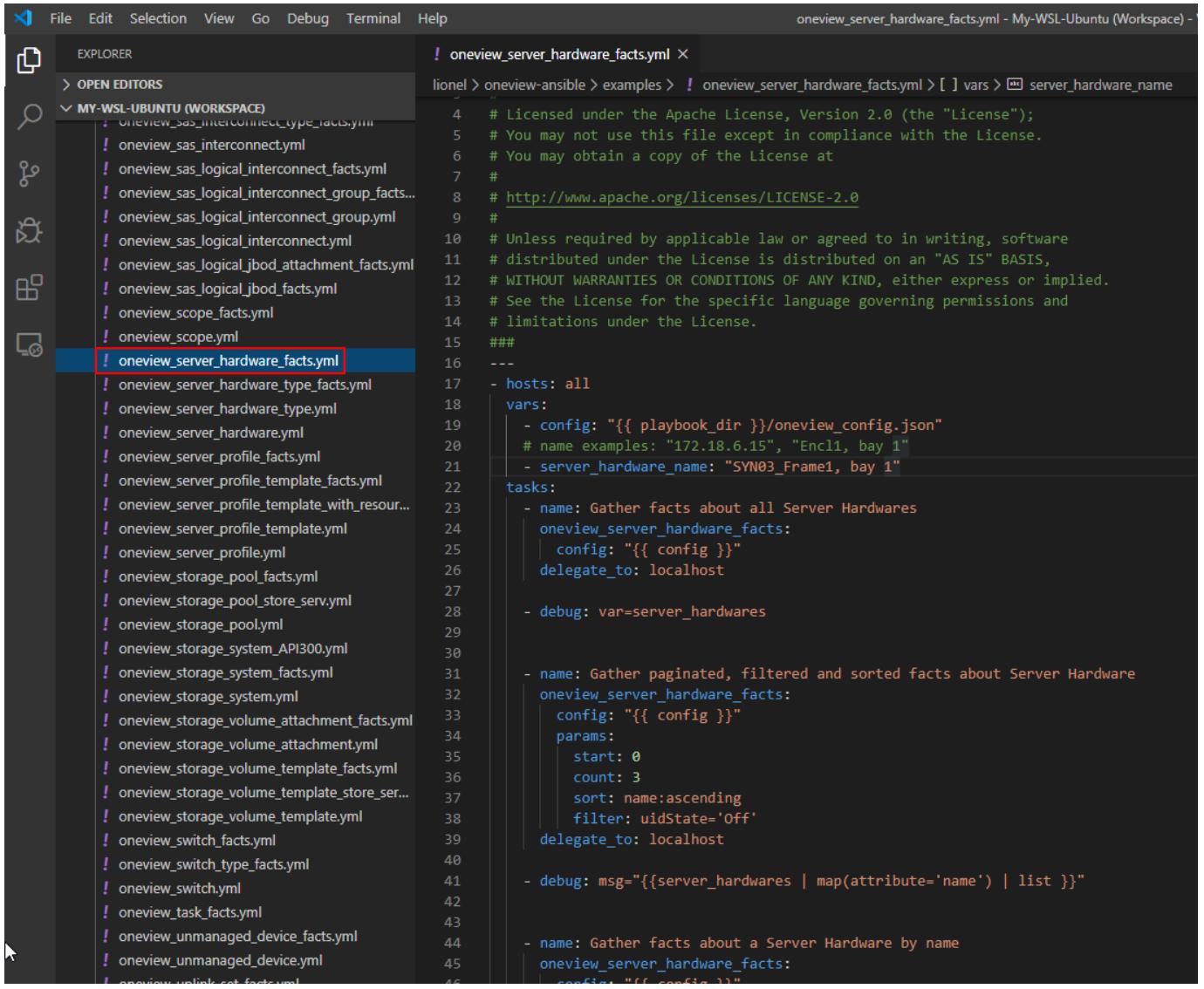
# Hewlett Packard Enterprise

- 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 X
lion@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 }}"
47
```



# Hewlett Packard Enterprise

- 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
```

1

2

3

4

**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.



# Hewlett Packard Enterprise

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

The screenshot shows a terminal window with the title "oneview\_server\_hardware\_facts.yml - My-WSL-Ubuntu (Works)". The left pane is an "EXPLORER" view showing a file tree. In the "MY-WSL-UBUNTU (WORKSPACE)" folder, there is a "library" folder which is expanded, revealing several Python files: module\_utils, \_\_init\_\_.py, hpe\_icsp\_os\_deployment.py, hpe\_icsp\_server.py, image\_streamer\_artifact\_bundle\_facts.py, image\_streamer\_artifact\_bundle.py, image\_streamer\_build\_plan\_facts.py, image\_streamer\_build\_plan.py, image\_streamer\_deployment\_group\_facts.py, image\_streamer\_deployment\_plan\_facts.py, image\_streamer\_deployment\_plan.py, image\_streamer\_golden\_image\_facts.py, image\_streamer\_golden\_image.py, image\_streamer\_os\_volume\_facts.py, image\_streamer\_plan\_script\_facts.py, and image\_streamer\_plan\_script.py. The right pane displays the content of the file "oneview\_server\_hardware\_facts.yml". The file contains Ansible tasks for gathering facts about server hardware. The "library" folder in the Explorer is highlighted with a red box.

```
! oneview_server_hardware_facts.yml ×
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 X
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),
43             | C(remoteConsoleUrl), 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.



# Hewlett Packard Enterprise

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

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

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

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

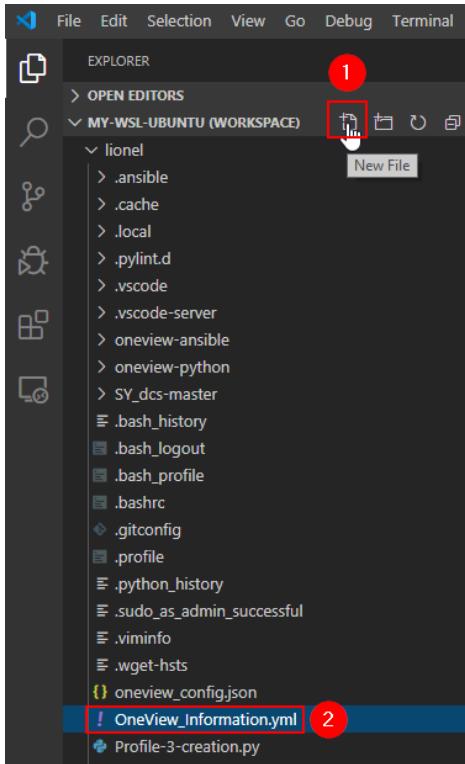
```
- debug: var=server_hardwares
```

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



# Hewlett Packard Enterprise

- Now let's create a new file in your user home directory named **OneView\_Information.yml**



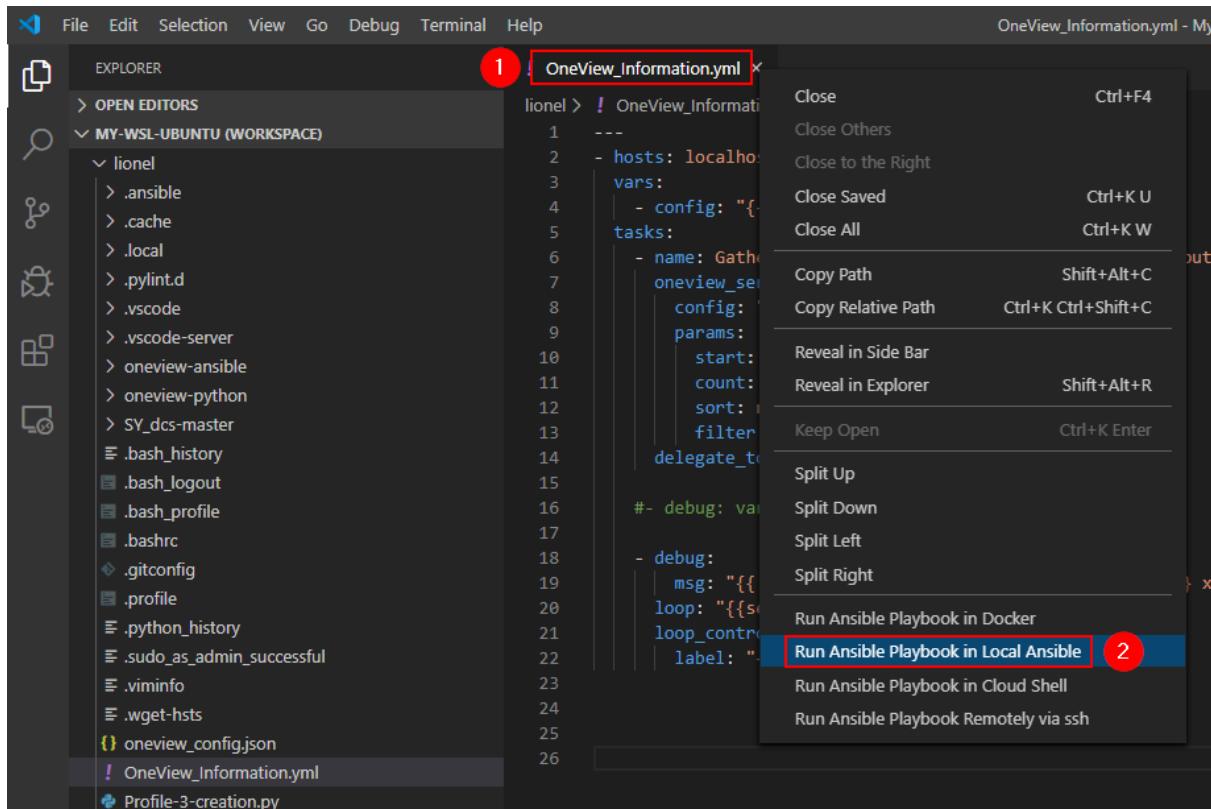
- Then copy/paste the following content:

```
---
- 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"
```



# Hewlett Packard Enterprise

- 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
```

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 }}"
```



# Hewlett Packard Enterprise

- Save the file then run it:

The screenshot shows a terminal window with tabs for OUTPUT, TERMINAL (which is selected), DEBUG CONSOLE, and PROBLEMS. The terminal displays the output of an Ansible play. It starts with a TASK [Gather paginated, filtered and sorted facts about Server Hardware] which lists several resources with their details. This is followed by a TASK [debug] which prints a list of network names. Finally, a PLAY RECAP summary is shown at the bottom.

```
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

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": [
        "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
```

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



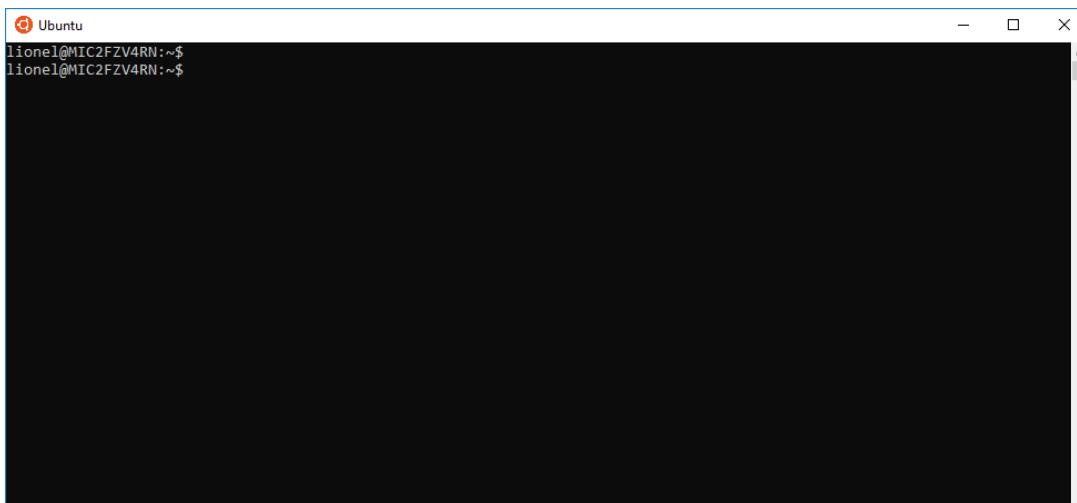
# Hewlett Packard Enterprise

## Ansible – Scenario 2 – Provisioning New Servers

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

### Prerequisites:

- Make sure your Ubuntu session running in WSL is started.



- 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 files, 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*.

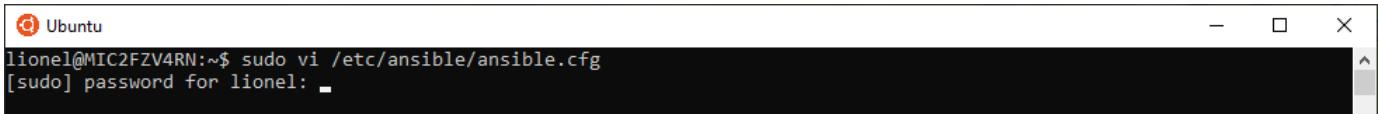


# Hewlett Packard Enterprise

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

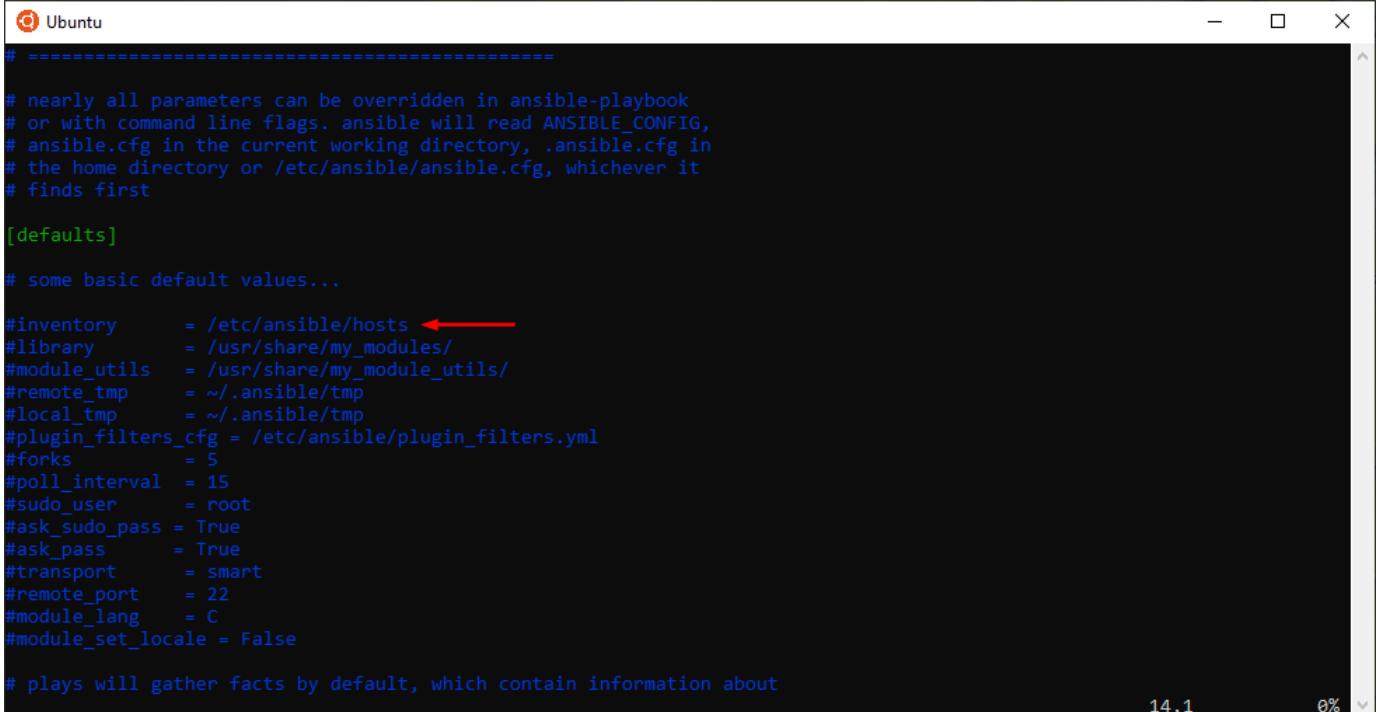
- Go to the WSL – Ubuntu console and enter:

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



```
Ubuntu
lionel@MIC2FZV4RN:~$ sudo vi /etc/ansible/ansible.cfg
[sudo] password for lionel: -
```

- Enter your password.
- Uncomment the inventory value:



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

[defaults]

# some basic default values...

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

# plays will gather facts by default, which contain information about
```

14,1

0%

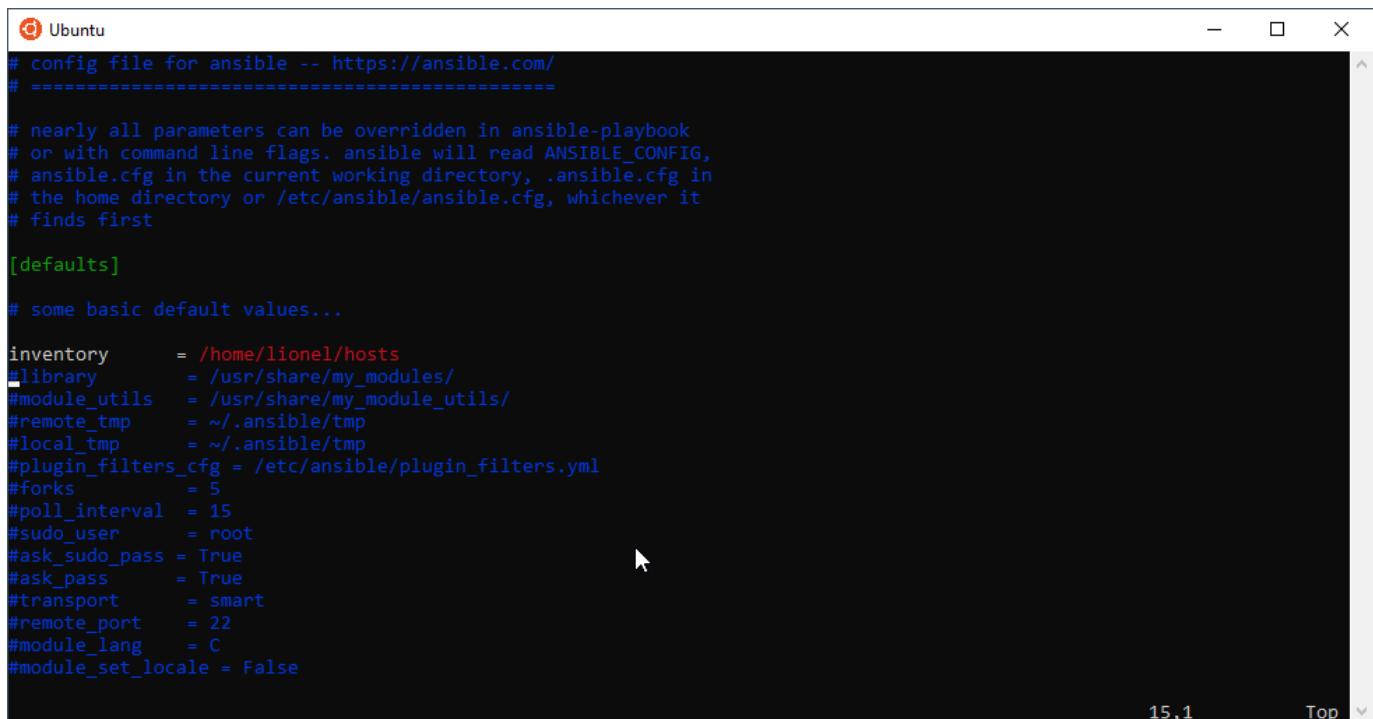


# Hewlett Packard Enterprise

- 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
```

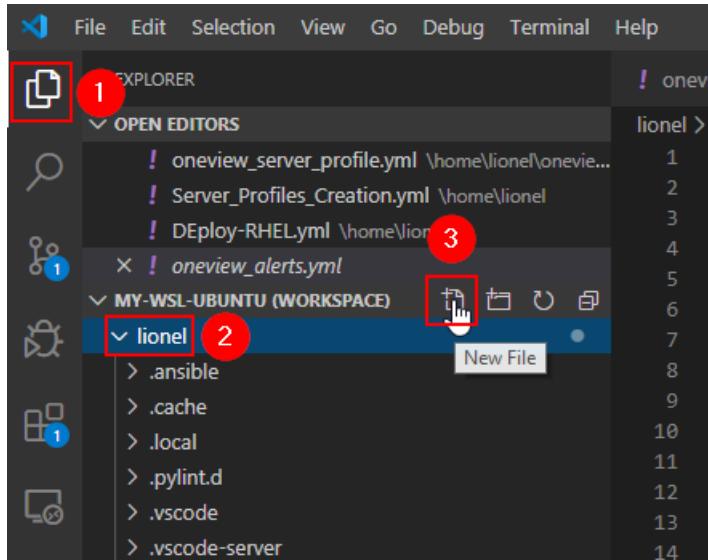
- Press **ESC** to exit *Insert Mode*, then type **:**(colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi

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

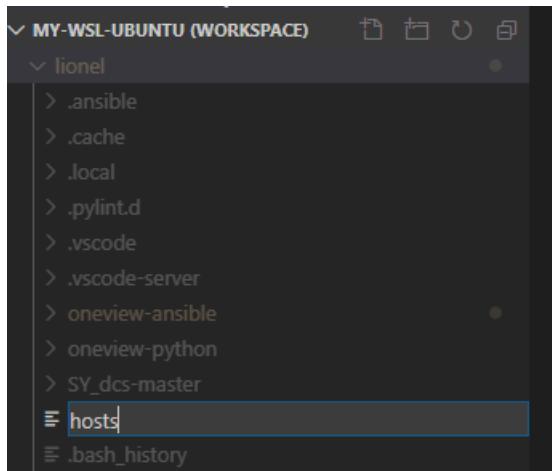
- Open VS Code, in the **Explorer** view, select your home folder then click on **New File**



# Hewlett Packard Enterprise



- Named it **hosts**:



- Then add in the hosts file the following content:

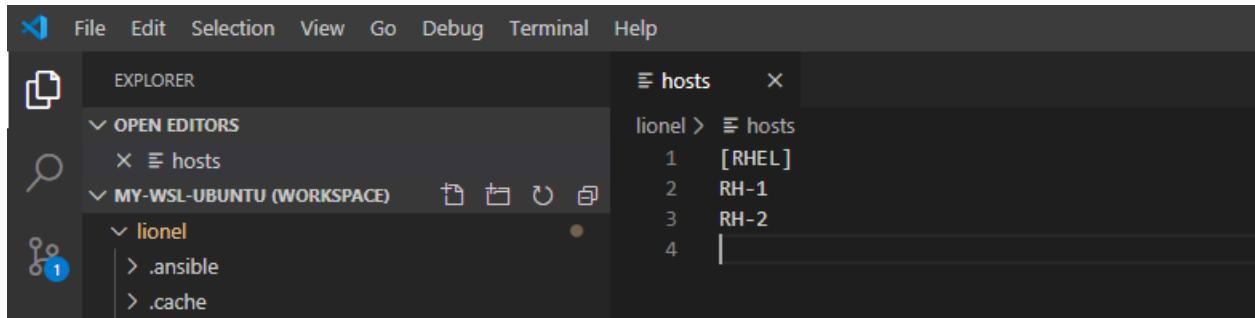
```
[RHEL]
RH-1
RH-2
```

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



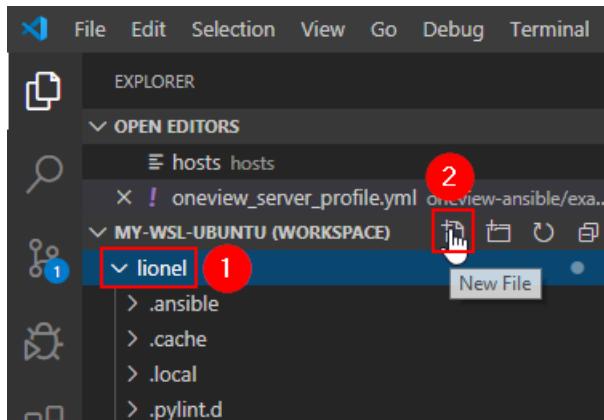
# Hewlett Packard Enterprise

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

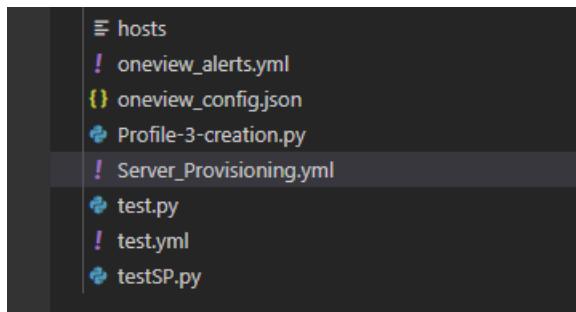


Now let's build a new playbook that will use this `hosts` file to create two server profiles (RH-1 and RH-2) from an existing Server Profile Template.

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



- Enter the name **Server\_Provisioning.yml**





# Hewlett Packard Enterprise

- Copy/paste the following playbook content:

```
---
- 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 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**



# Hewlett Packard Enterprise

This playbook contains two main tasks, the first one creates the server profiles:

- ①: Creates 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 playbook.
- ②: Defines the Server Profile Template (SPT) that Ansible must use to generate the Server Profiles (SP), the SPT is set in line 7.
- ③: Tells Ansible to use the hosts inventory file to generate the SP names.
- ④: Displays the result of the SP creation task

```
---
```

```
- name: Ansible OneView Synergy playbook to deploy Compute Module(s) using a Server Profile Template
  hosts: RHEL
  gather_facts: no
  vars:
    - config: "{{ playbook_dir }}/oneview_config.json"
    - server_template: "HPE Synergy 480 Gen9 with Local Boot for RHEL Template"

  tasks:
    - name: Creating Server Profile [{{ inventory_hostname }}] from Server Profile Template [{{ server_template }}]
      oneview_server_profile: ← 1
        config: "{{ config }}"
        data:
          serverProfileTemplateName: "{{ server_template }}"
          name: "{{ inventory_hostname }}" ← 2
      delegate_to: localhost
      register: result

      #- debug: var=server_hardware

      - name: Task result of the Server Profile(s) creation
        debug:
          msg: "{{ result.msg }}" ← 3
    - name: Task result of the Server Profile(s) creation
      debug:
        msg: "{{ result.msg }}" ← 4
```

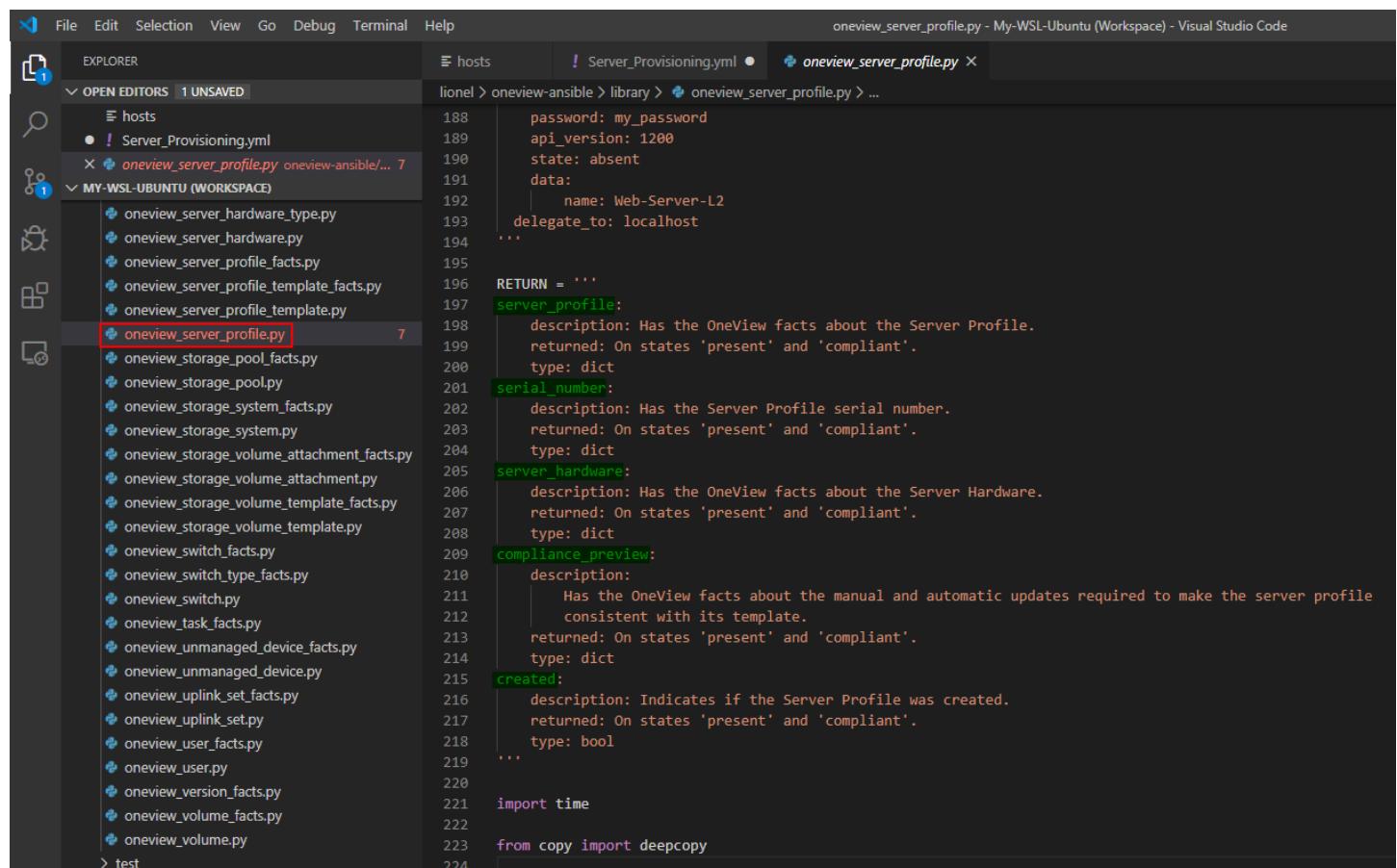
# Hewlett Packard Enterprise

Then in the second task, 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 }}]
  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 }}"
```

As described below in the `oneview_server_profile` module documentation located in `/oneview-ansible/library/oneview_server_profile.py`, `oneview_server_profile` returns 4 variables, `server_profiles`, `serial_number`, `server_hardware`, `compliance_preview` and `created`.



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

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Explorer:** Shows the file structure:
  - OPEN EDITORS: Server\_Provisioning.yml (marked with a red circle), oneview\_server\_profile.py (marked with a red circle).
  - MY-WSL-UBUNTU (WORKSPACE): A list of Python files under the oneview-ansible library, including oneview\_server\_hardware\_type.py, oneview\_server\_hardware.py, oneview\_server\_profile\_facts.py, oneview\_server\_profile\_template\_facts.py, oneview\_server\_profile\_template.py, and oneview\_server\_profile.py (highlighted with a red box).
- Code Editor:** The content of the `oneview_server_profile.py` file is displayed, showing various Ansible tasks and variables. The variables defined include:
  - `password: my_password`
  - `api_version: 1200`
  - `state: absent`
  - `data:` which contains:
    - `name: Web-Server-L2`
    - `delegate_to: localhost`
    - `...`
  - `RETURN = ...`
  - `server_profile:` which includes:
    - `description: Has the OneView facts about the Server Profile.`
    - `returned: On states 'present' and 'compliant'.`
    - `type: dict`
  - `serial_number:` which includes:
    - `description: Has the Server Profile serial number.`
    - `returned: On states 'present' and 'compliant'.`
    - `type: dict`
  - `server_hardware:` which includes:
    - `description: Has the OneView facts about the Server Hardware.`
    - `returned: On states 'present' and 'compliant'.`
    - `type: dict`
  - `compliance_preview:` which includes:
    - `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:` which includes:
    - `description: Indicates if the Server Profile was created.`
    - `returned: On states 'present' and 'compliant'.`
    - `type: bool`
  - `import time`
  - `from copy import deepcopy`



# Hewlett Packard Enterprise

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**

```
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
hosts \home\lionel
Server_Provisioning.yml
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
.profile
.python_history
.sudo_as_admin_successful
.viminfo
.wget-hsts
hosts
lionel > Server_Provisioning.yml >
1 --- 
2 - name: Ansible OneView Provisioning
  hosts: RHEL
  gather_facts: no
  vars:
    - config: "{{ playbooks_dir }}/{{ inventory_file }}"
    - server_template: "OneView"
  tasks:
    - name: Creating OneView Server Profile
      oneview_server_profile:
        config: "{{ config }}"
        data:
          serverProfileName: "{{ inventory_hostname }}"
        delegate_to: localhost
        register: results
    - name: Task results
      debug:
        msg: "{{ results }}"
    - name: Powering On OneView Server
      oneview_server_power:
        config: "{{ config }}"

```



# Hewlett Packard Enterprise

- Open the web address <https://192.168.56.101/#/profiles> to show the creation of the two server profiles:

The screenshot shows the HPE OneView interface for managing server profiles. On the left, a sidebar lists four server profiles: Profile-1, Profile-2, RH-1, and RH-2. The RH-1 profile is currently selected and highlighted with a green background. The main content area displays the 'Overview' of the RH-1 profile. At the top of this area, there is a progress bar labeled 'Create' and a note stating 'Apply settings to Synergy-Encl-1,bay,4.' Below this, the 'General' section provides detailed configuration information for the profile, including:

Setting	Value
Description	Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	Synergy-Encl-1,bay,4
Server hardware type	SY 660 Gen9
Enclosure group	EG-Synergy-Local
Affinity	Device bay
Server power	On
Serial number (v)	VCGOTIV002
UUID (v)	05d42c4e-557d-448e-aa9e-ea11985e9f97
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgotiv002

Below the General section, there is a 'Firmware' section which indicates the firmware baseline is managed manually.



# Hewlett Packard Enterprise

- Open the VS Code terminal to see the outputs of our Ansible playbook tasks:

```
OUTPUT TERMINAL 1 BUG CONSOLE PROBLEMS

PLAY [Ansible OneView Synergy playbook to deploy Compute Module(s) using a Server Profile Template] ****
skipping: no hosts matched

PLAY RECAP ****

lione1@MIC2FZV4RN:~$ export VSCODEEXT_USER_AGENT=vscozz.vscode-ansible-0.5.2
lione1@MIC2FZV4RN:~$ ansible-playbook "/home/lione1/HPE-Synergy-demonstration-environment/Ansible/Server_Provisioning.yml"

PLAY [Ansible OneView Synergy playbook to deploy Compute Module(s) using a Server Profile Template] ****

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-1 -> localhost]
changed: [RH-2 -> localhost]

TASK [Task result of the Server Profile(s) creation] ****
ok: [RH-1] => {
    "msg": "Server Profile created."
}
ok: [RH-2] => {
    "msg": "Server Profile created."
} 2

TASK [Powering on the Compute Module(s) [Synergy-Encl-1, bay 4]] ****
changed: [RH-2 -> localhost] 3
changed: [RH-1 -> localhost]

TASK [debug] ****
ok: [RH-1] => {
    "msg": "The server is located in Synergy-Encl-1, bay 4"
}
ok: [RH-2] => {
    "msg": "The server is located in Synergy-Encl-2, bay 4" 4
}

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

lione1@MIC2FZV4RN:~$
```

- ②: Task result of the server profiles creation.
- ③: Task result of the powering-on of the server hardware.
- ④: Displays the server hardware location used by the server profiles



# Hewlett Packard Enterprise

- 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 4" with a "Create profile" button. Below it, a list shows four profiles: "Name" (radio button), "Profile-1" (green dot), "Profile-2" (green dot), "RH-1" (green dot, highlighted with a green box), and "RH-2" (green dot, highlighted with a red box). The main panel shows the details for "RH-1". It has tabs for "Overview" and "Actions". Under "General >", there are several configuration items with their values:

Setting	Value
Description	Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	Synergy-Encl-1, bay 4
Server hardware type	SY 660 Gen9.1
Enclosure group	EG-Synergy-Local
Affinity	Device bay
Server power	On
Serial number (v)	VCGOTIV002
UUID (v)	05d42c4e-557d-448e-aa9e-ea11985e9f97
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgotiv002

This concludes Ansible - Scenario 2



## Ansible – Scenario 3 – Unprovisioning Running Servers

**Note:** This scenario can only be run after *Ansible – Scenario 2* when server profiles *RH-1* and *RH-2* are available in OneView.

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.

- 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.



# Hewlett Packard Enterprise

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

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

- File Tab Context Menu:** A context menu is open over the "Server\_Unprovisioning.yml" file tab in the title bar. Item 1, "Run Ansible Playbook in Local Ansible", is highlighted with a red circle.
- Explorer View:** On the left, the "EXPLORER" view shows a file tree with several files and folders under "MY-WSL-UBUNTU (WORKSPACE)" and "lionel".
- Code Editor:** The main editor area displays the content of the "Server\_Unprovisioning.yml" playbook. The code uses Ansible syntax to manage server profiles, including tasks for gathering facts, getting server profile information, powering off hardware, and deleting profiles.
- Context Menu Options:** The context menu lists various options for managing the file, including "Run Ansible Playbook in Docker", "Run Ansible Playbook in Cloud Shell", and "Run Ansible Playbook Remotely via ssh". Item 2, "Run Ansible Playbook in Local Ansible", is also present in this list.



# Hewlett Packard Enterprise

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

Server Profiles 4

+ Create profile

Name

- Profile-1
- Profile-2
- RH-1
- RH-2

General >

Description	Server Profile for HPE Synergy 660 Gen9 Compute Module with Local Boot and SAN Storage for Windows
Server profile template	<a href="#">HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template</a>
Server hardware	Synergy-Encl-1,bay.4
Server hardware type	SY 660 Gen9.1
Enclosure group	EG-Synergy-Local
Affinity	Device bay
Server power	Off
Serial number (v)	VCGOTIV002
UUID (v)	05d42c4e-557d-448e-aa9e-ea11985e9f97
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcgotiv002

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

Server Profiles 0

+ Create profile

Name

- RH-1

Completed 28s

administrator 2/25/20 11:13:57 am



# Hewlett Packard Enterprise

- In the terminal, the following outputs get displayed:

```
lione1@MIC2FZV4RN:~$ ansible-playbook "/home/lionel/HPE-Synergy-demonstration-environment/Ansible/Server_Unprovisioning.yml"

PLAY [Ansible OneView Synergy playbook to remove deployed servers] *****

TASK [Getting server profile(s) information] *****
ok: [RH-2 -> localhost]
ok: [RH-1 -> localhost]

TASK [Powering off the server hardware [Synergy-Encl-1, bay 4]] *****
changed: [RH-2 -> localhost] 1
changed: [RH-1 -> localhost] 2

TASK [Deleting Server Profile [RH-1]] *****
changed: [RH-2 -> localhost]
changed: [RH-1 -> localhost] 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

lione1@MIC2FZV4RN:~$
```

- ①: Task result of the server hardware powering-off.
- ②: Task result of the server profiles deletion.

This concludes Ansible – Scenario 3



## Hewlett Packard Enterprise

We have reached the end of our scenarios to demonstrate a Synergy Infrastructure programmability, infrastructure as-code and all the good benefits and features of our Unified API.

In the next chapter, we are going to reset our environment to prepare our next customer visit.



## Chapter-8 - How to reset the Synergy demonstration environment

We have successfully completed all our live demos and we need now to reset our environment to be ready for our next customer live demonstrations.

The procedure is simple as we have prepared two snapshots to return easily and quickly to a previous state of our DCS appliance. As a reminder, each snapshot provides a different use case:

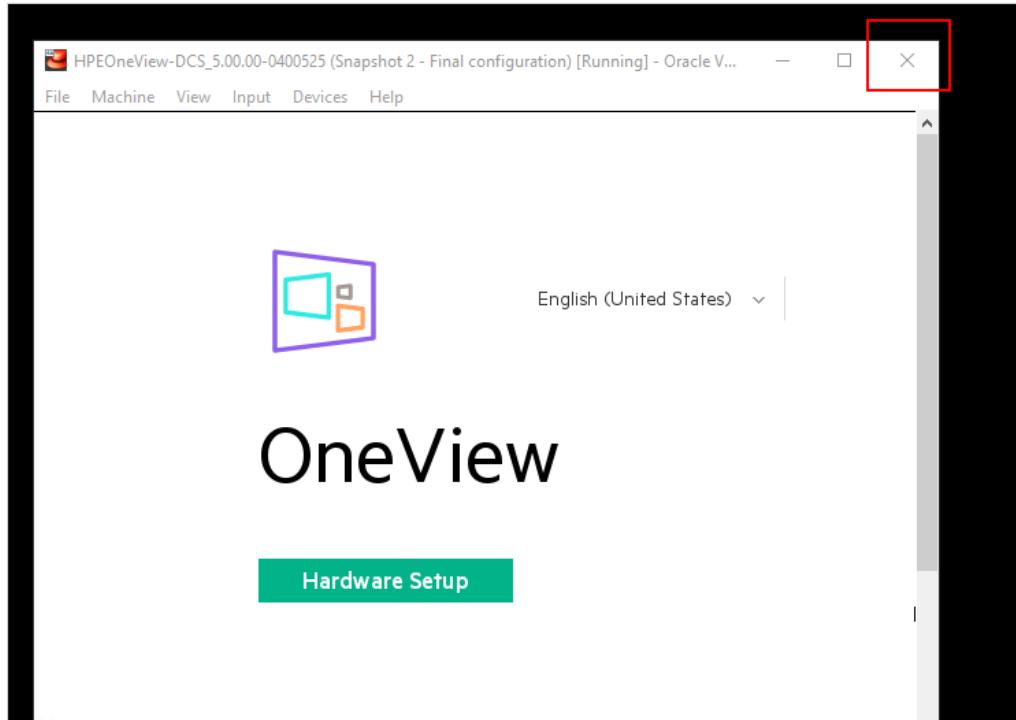
- 1- First snapshot: OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network)
  - ⇒ Can be used to show how to automate the setup of Synergy and the power of our infrastructure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

## Shutting down the Synergy Composer Demonstration appliance

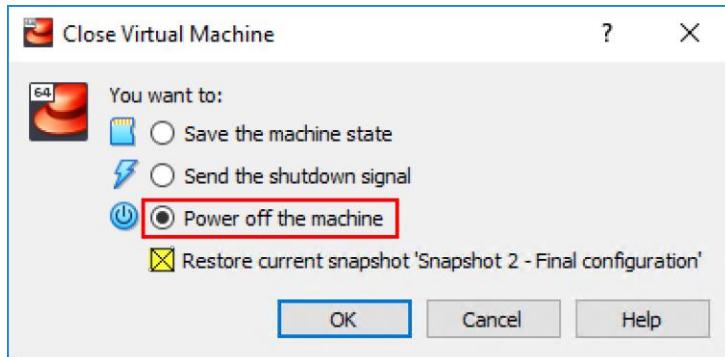
With VirtualBox, when shutting down a virtual machine, you have a "Restore current snapshot" option if you want VirtualBox to load the most recent snapshot the next time you start up the virtual machine again. This is a very convenient and easy option to restore the appliance back to the pre-configured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

To restore the appliance back to a pre-demo state:

- Click on the **Close** button in the upper-right corner of the window of the DCS appliance virtual machine



- Select **Power off the machine** and check the restore current snapshot option to restore the **Final configuration** snapshot the next time you start the appliance:



- Click the **OK** button to begin the shutdown process.

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

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

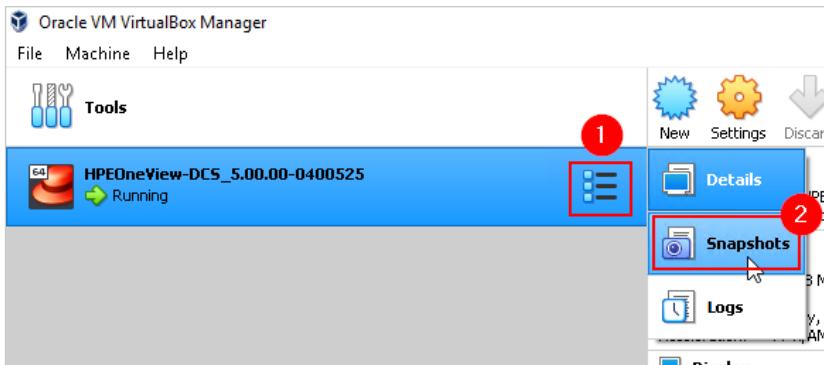


# Hewlett Packard Enterprise

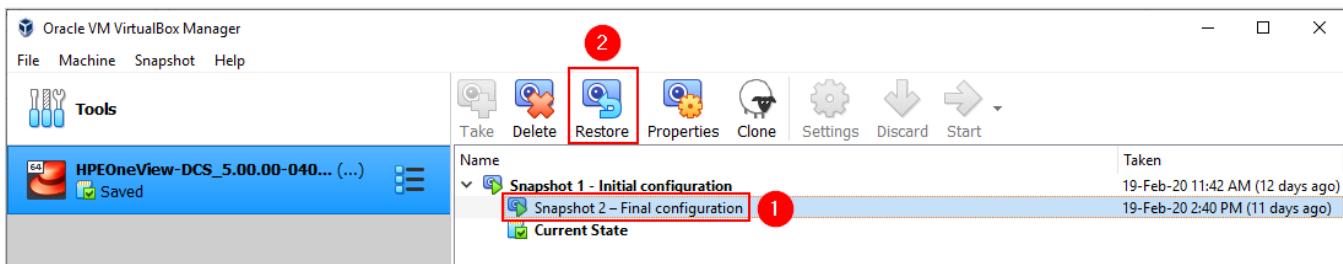
## Resetting the Synergy Composer Demonstration appliance to the fully configured state

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.

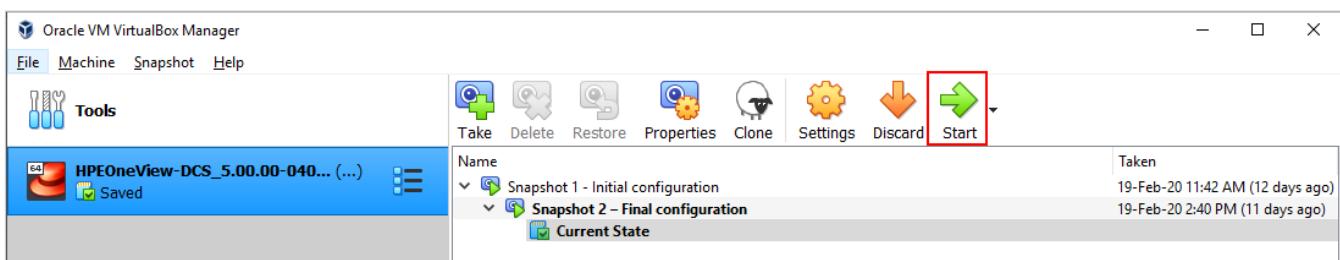
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- 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.

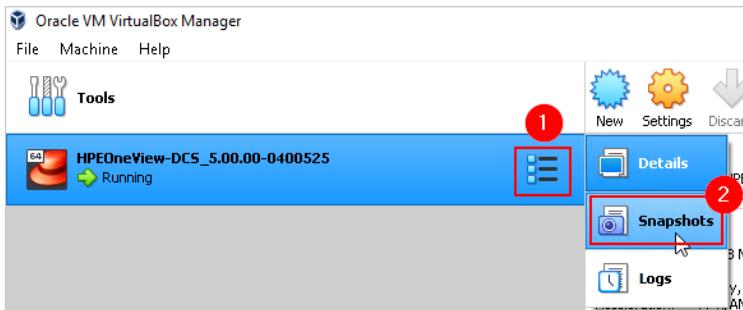


# Hewlett Packard Enterprise

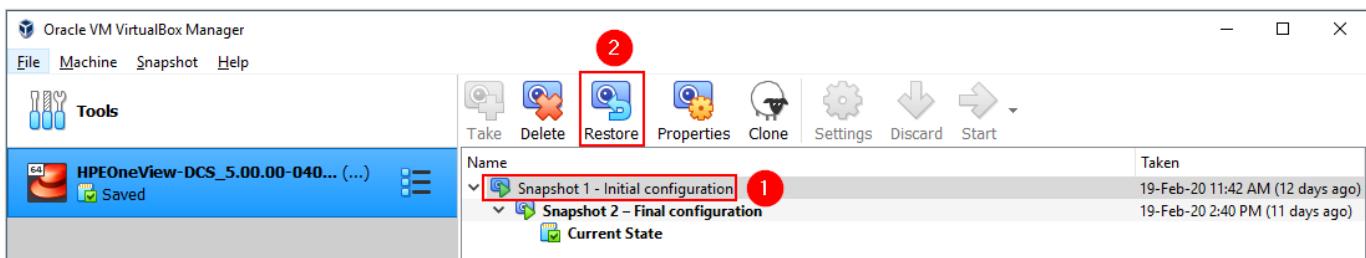
## Resetting the Synergy Composer Demonstration appliance to the unconfigured state

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.

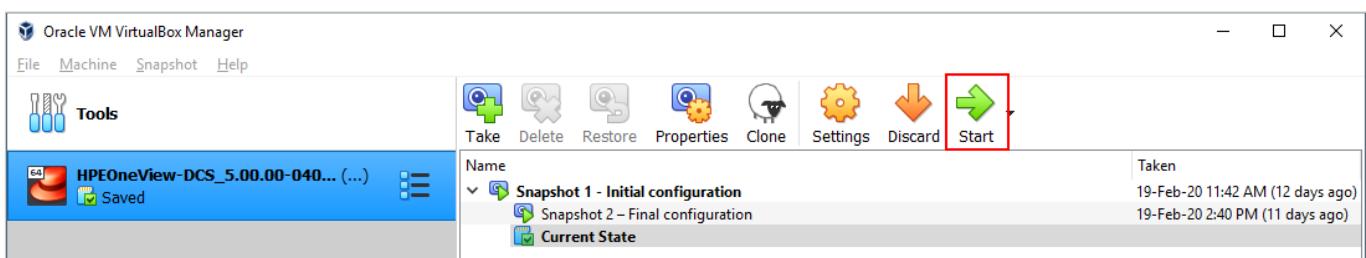
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- 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.

This concludes Chapter-8



## Troubleshooting

### 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.

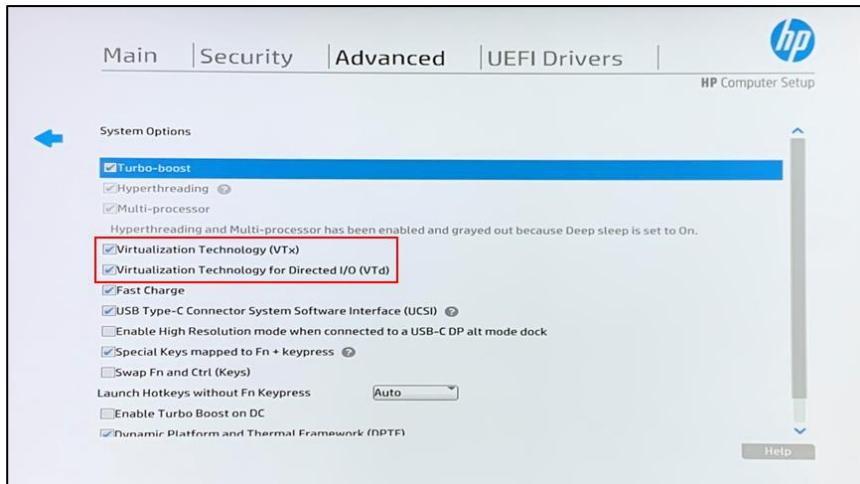


# Hewlett Packard Enterprise

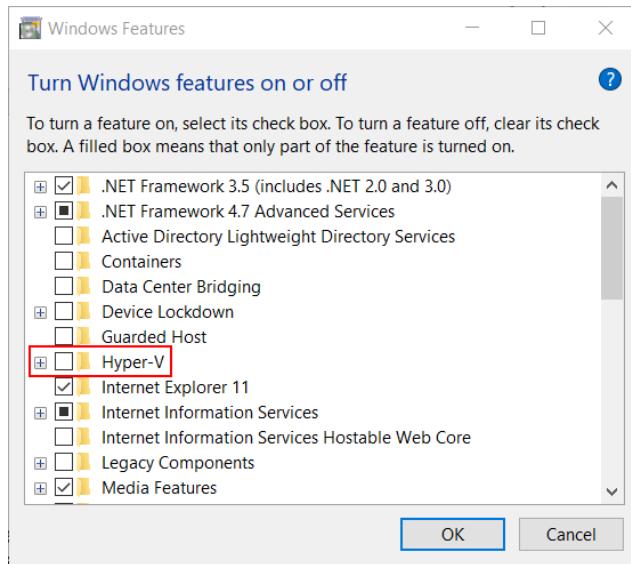
## Solution:

- 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:

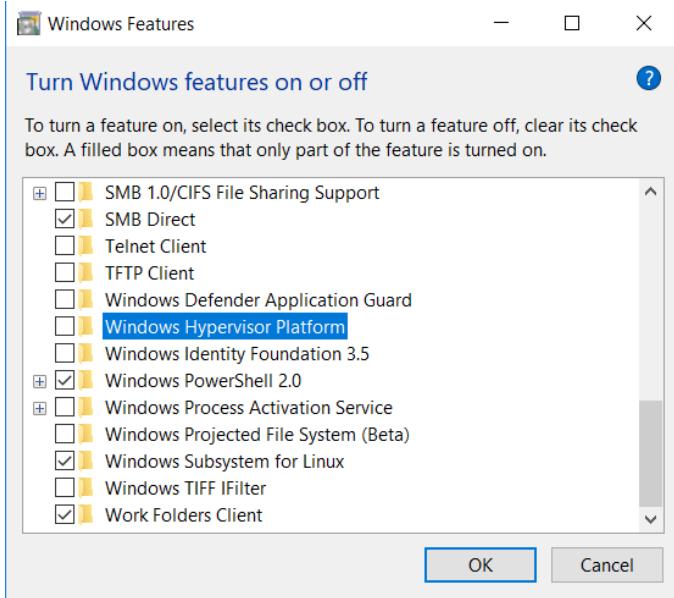
```
dism.exe /Online /Disable-Feature:Microsoft-Hyper-V
```

**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



# Hewlett Packard Enterprise

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



Or run:

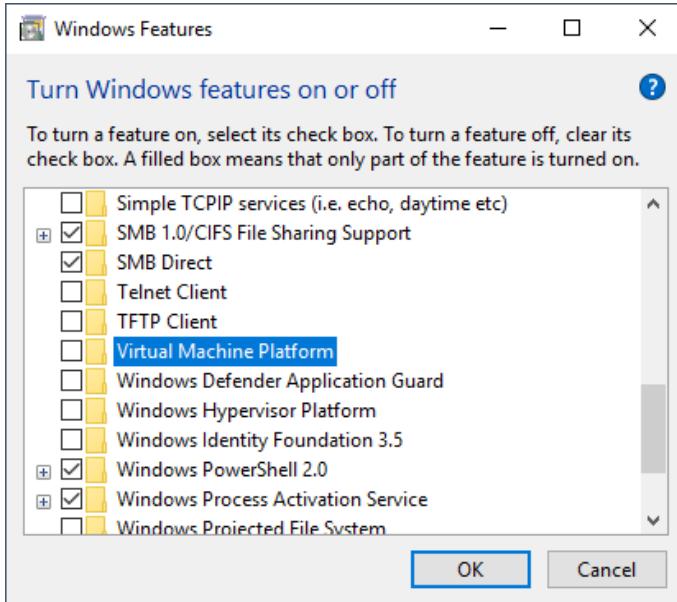
```
dism.exe /Online /Disable-Feature:HypervisorPlatform
```

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



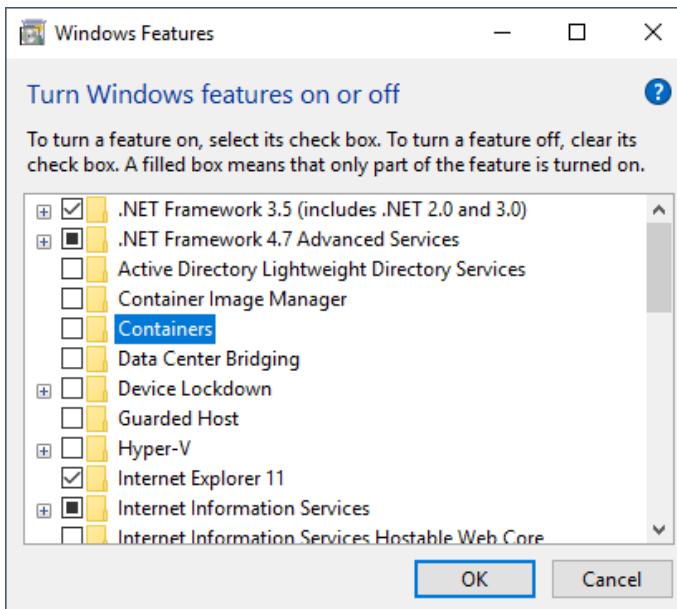
# Hewlett Packard Enterprise

- 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).

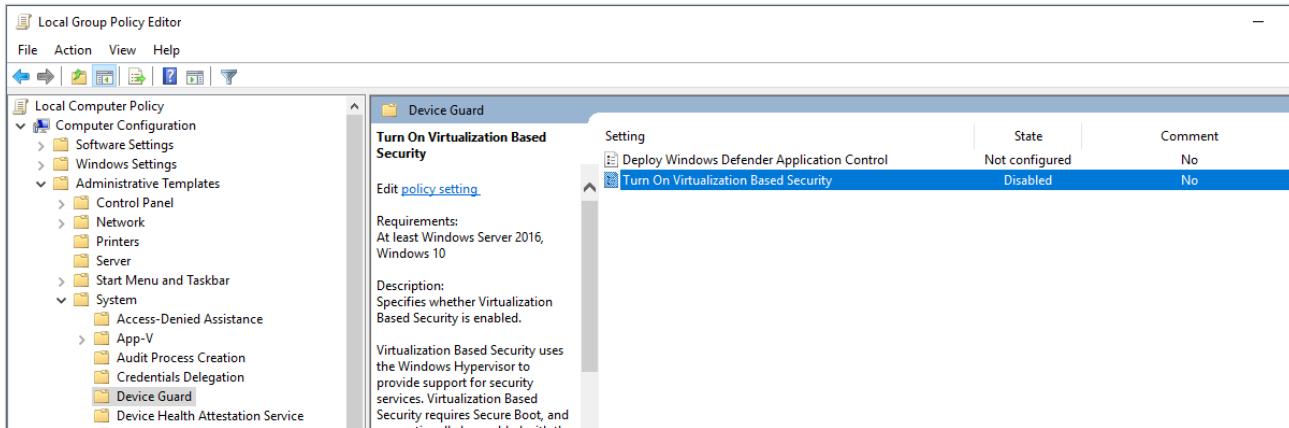


**Note:** If running, *Container* is taking exclusive use of VT-x so it is required to turn it off



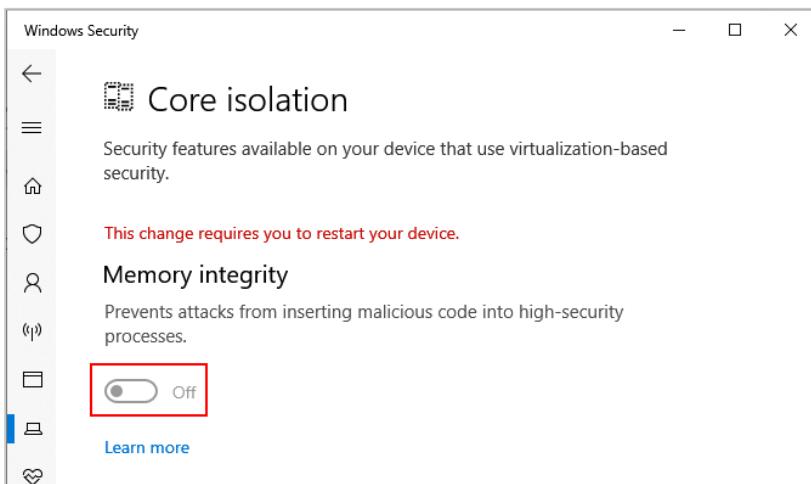
# Hewlett Packard Enterprise

- 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"





# Hewlett Packard Enterprise

## **Hardware and Software versions used to build this demonstration environment**

Major components used in this paper are listed here. The list is not exhaustive and is only intended to provide guidance.

Hardware description:

- HP EliteBook 840 G5
- 16G 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.

Software versions:

- Windows 10 version 1809 (OS Build 17763.1039)
- Visual Studio Code version 1.42.1
- Oracle VM VirtualBox version 6.1.4 r136177 (Qt5.6.2)
- HPE OneView DCS 5.00 Z7550-96681
- Ubuntu 20190521 build on 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
- Ansible 2.9.4
- PowerShell for OneView library 5.0.2341.1920



# Hewlett Packard Enterprise

## Summary

The HPE Synergy demonstration environment installation is complete.

Today's Idea Economy is driving IT leaders to find new and innovative ways to deliver the flexibility of hybrid IT solutions while reducing complexity and operating costs. Composable Infrastructure provides the promise of allowing IT to provision and manage traditional workloads along with new mobile and cloud-native applications from a single infrastructure, allowing you to achieve the vision of infrastructure as a code.

You have now the environment to seamlessly prove this and demonstrate the power of a Synergy Composable Infrastructure!

And remember, a product demonstration is one of your best tools to strongly impact your customers to adopt new technologies!

Happy demonstrations!

To help us improve this document, please provide feedback at [lio@hpe.com](mailto:lio@hpe.com).