Hewlett Packard Enterprise

Technical white paper

Building an HPE Synergy demonstration environment on your own laptop

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

Version: July 2020

Document Revision History

Publish / Revised	Software versions	Section and Text Revised
June 2020	- HPE OneView DCS **5.20** Z7550-96879 - Windows 10 version **1909** (OS Build 18363.418) - Visual Studio Code version **1.45.1** - PowerShell **7.0.2** - Oracle VM VirtualBox version **6.1.8** r137981 (Qt5.6.2) - Windows Subsystem for Linux with Ubuntu 18.04 LTS - Ansible Modules for HPE OneView **5.6.0** with bug_fix/create_profiles_in_parallel - HPE OneView Python SDK **5.20** - Python 2.7.17 - 3.6.9 - Ansible **2.9.9** - PowerShell for OneView library **5.2.4343.1695**	- Update content for HPE Synergy OneView Demonstration appliance - DCS 5.20 - Added Terraform installation in WSL and in VS Code - Added Terraform provider for OneView and Go installation in WSL - Added 3 x Terraform scenarios - Added Chapter-9 - How to upgrade the Synergy demonstration environment - Script improvements for Ansible and Python - Added WSL and Linux distribution recommendations to prevent incompatibility issues - Other minor changes
March 2020	- HPE OneView DCS 5.00 Z7550-96681 - Windows 10 version 1809 (OS Build 17763.1039) - Visual Studio Code version 1.42.1 - PowerShell 5.0 - Oracle VM VirtualBox version 6.1.4 r136177 (Qt5.6.2) - Windows Subsystem for Linux with Ubuntu 18.04 LTS - Ansible Modules for HPE OneView 5.4.0 with create_profiles_in_parallel 11/27/19 fix - HPE OneView Python SDK 5.00 - Python 2.7.17 - 3.6.9 - Ansible 2.9.4 - PowerShell for OneView library 5.0.2341.1920	

Abstract

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

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

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

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

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

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

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

Requirements

- A powerful laptop with minimum 16G of RAM (best is 32G) and with Intel Virtualization Technology enabled
- CentOS 7.5 Boot ISO
- OneView DCS 5.20 OVA File(s) for Synergy or for BL/DL (the installation and setup procedures are the same for both).
- An update version of Windows 10 (preferably April 2018 update, version 1803) to run the latest version of Windows Subsystem for Linux

Demonstration appliance prerequisites

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

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

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

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

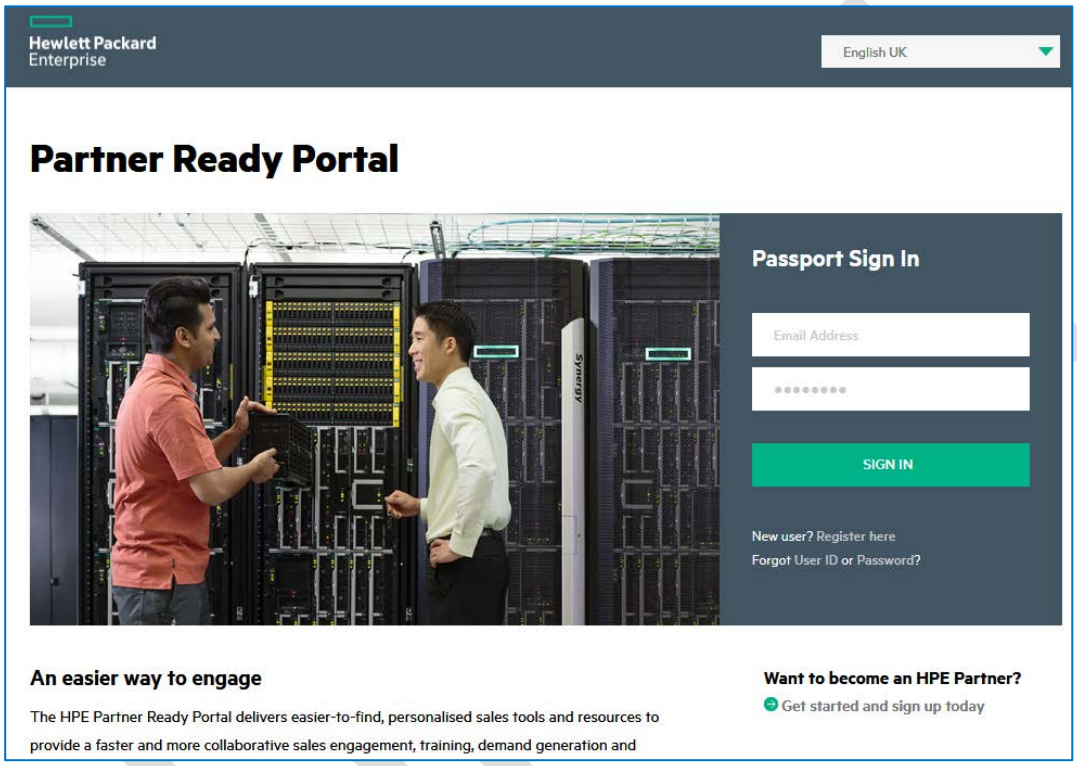
Note: The HPE OneView demonstration appliance is intended for demonstration purposes only and is only available to HPE employees and HPE Partners.

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

Downloading HPE OneView demonstration appliance

Downloading location for HPE Partners:

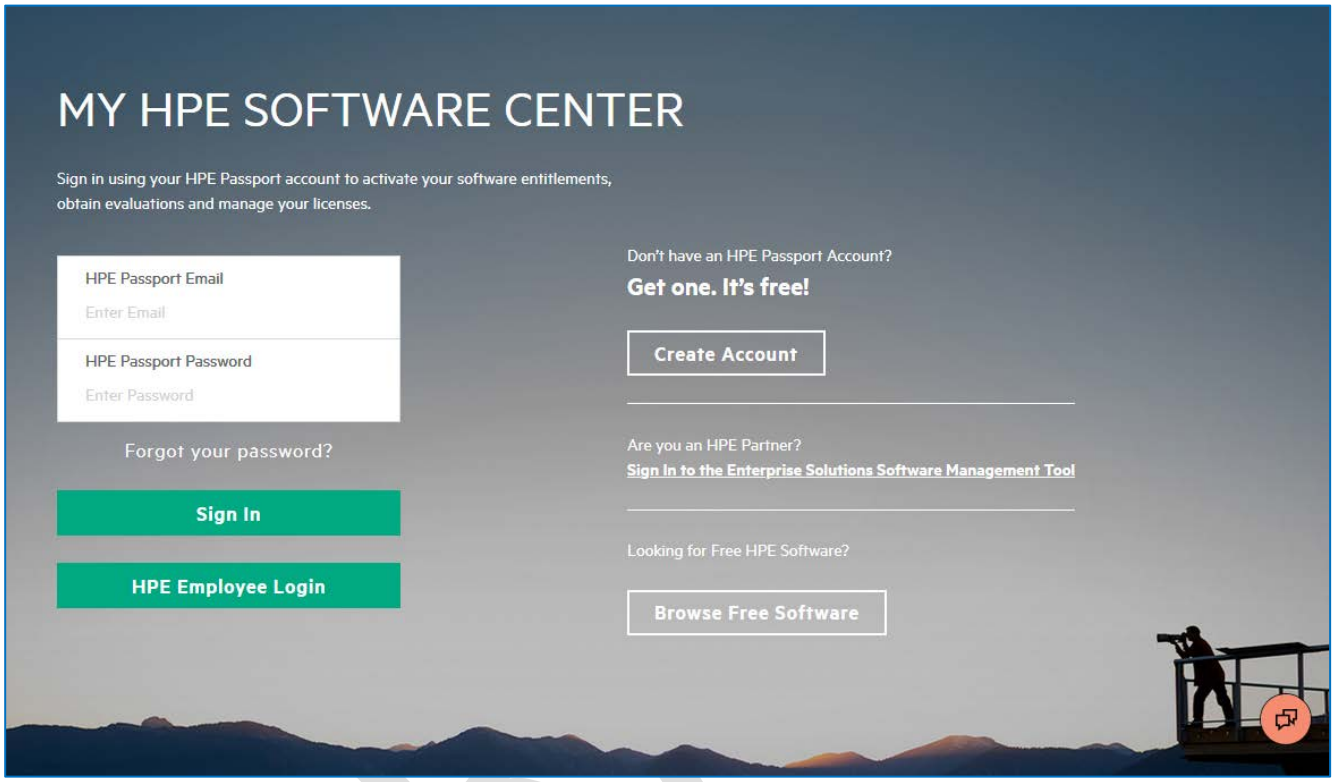
- Log in using your HPE Passport credentials to the *HPE Partner Ready Portal* https://partner.hpe.com/



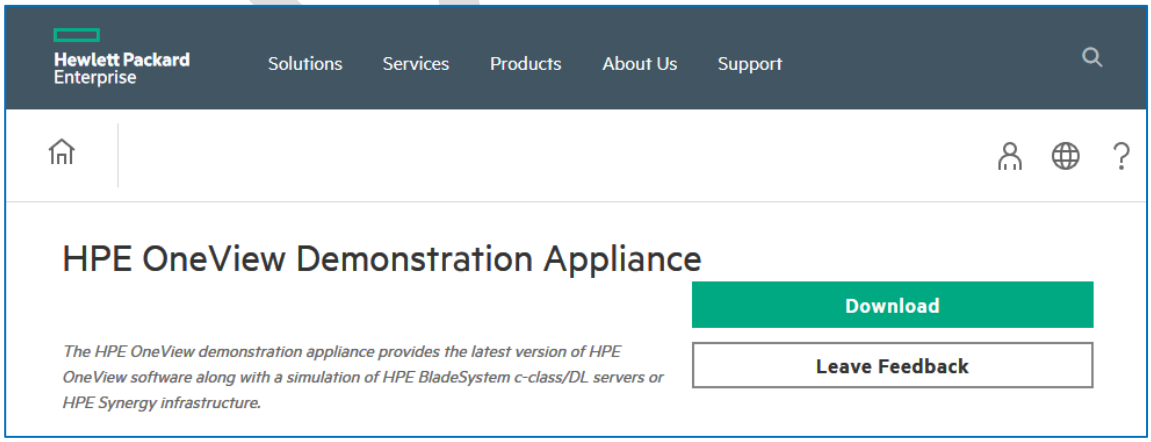
- Select **My Workspace** -> **Manage Software and Licenses**
- Select **SW evaluations**
- Select **All Categories** and scroll to **OneView**
- Accept the software terms and conditions
- Select **HPE_OneView_DCS_5.00_Synergy_ESXi_Z7550-96681.ova** then click **Download**

Downloading location for HPE Employees:

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

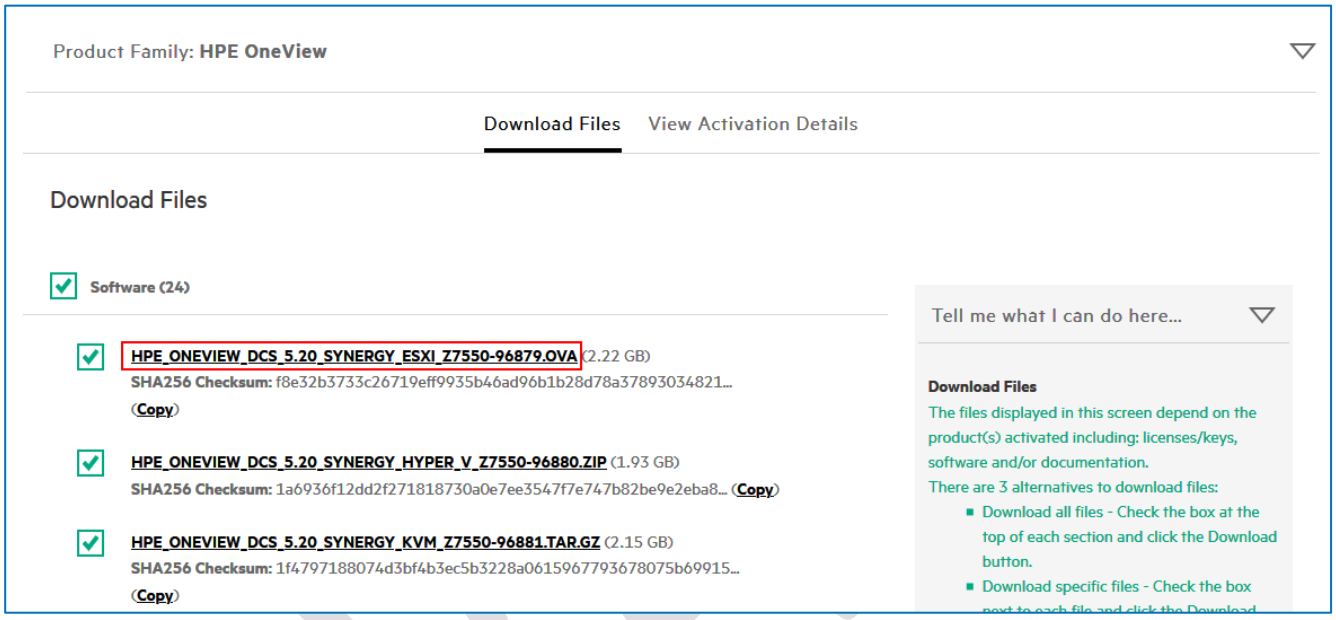


- Select **Software for HPE Employees**
- Select **OneView** in the Software Category drop down menu
- Scroll down and select **HPE OneView Demonstration Appliance**



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

- Accept the software terms and conditions
- Select **HPE_OneView_DCS_5.20_Synergy_ESXi_Z7550-96879.ova** then click **Download**



Downloading CentOS 7.5 Boot ISO

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

- Download the ISO from http://mirrors.usc.edu/pub/linux/distributions/centos/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso

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

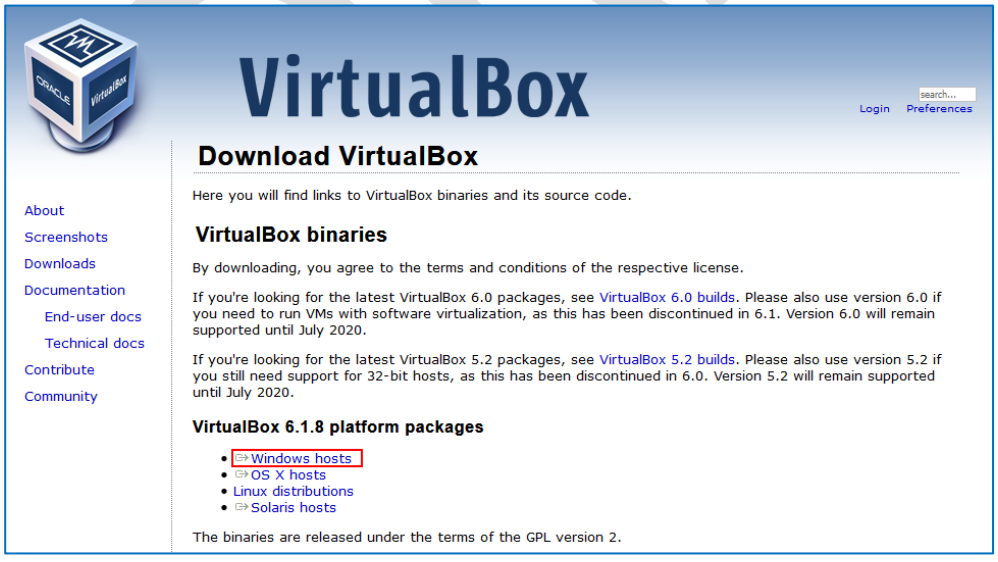
Downloading and installing VirtualBox on your PC

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

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

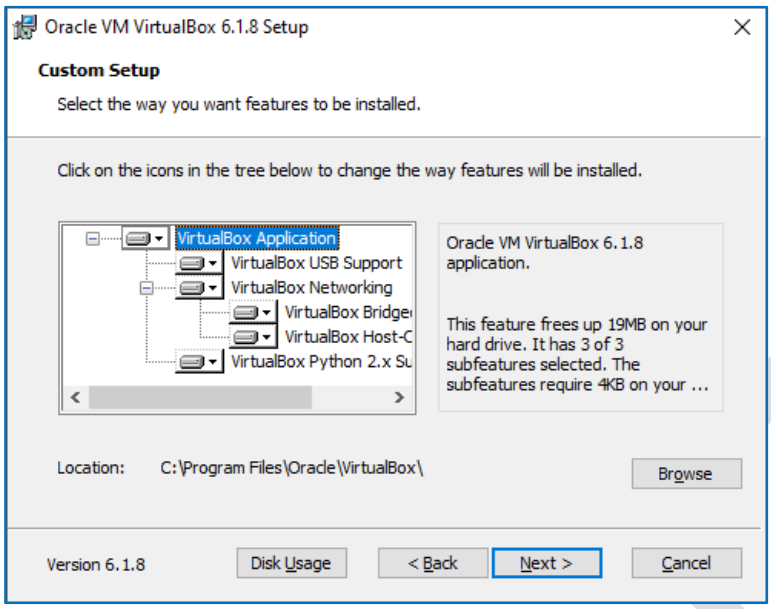
As mentioned in the prerequisites, the HPE OneView demonstration appliance is not officially supported with a VirtualBox hypervisor. The main reason for that is simply because it has not been tested and validated by HPE, but the experience shows that everything is working properly and as expected if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

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

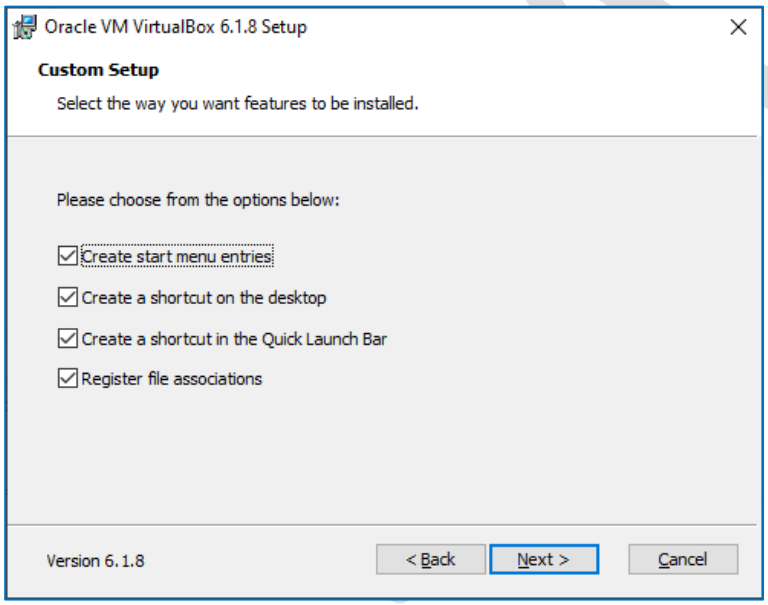


- Save the file on your PC and launch the executable.

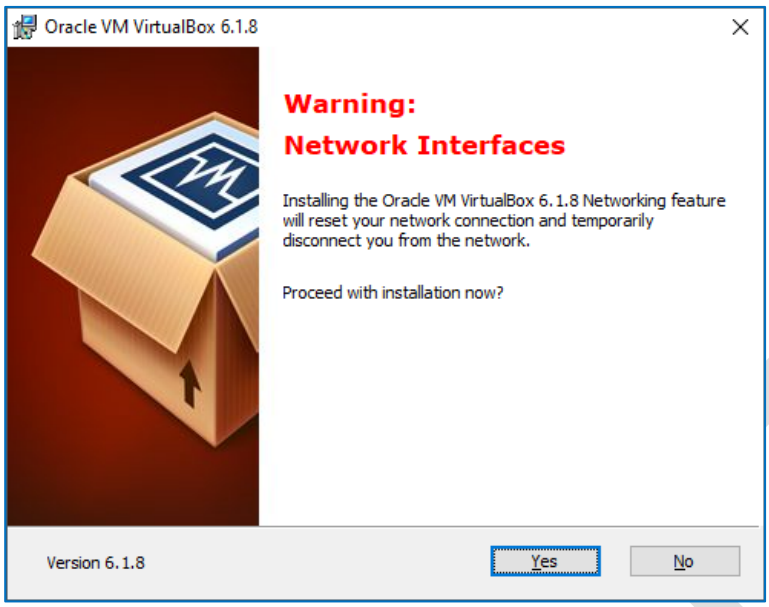
- Leave all default parameters then click **Next**



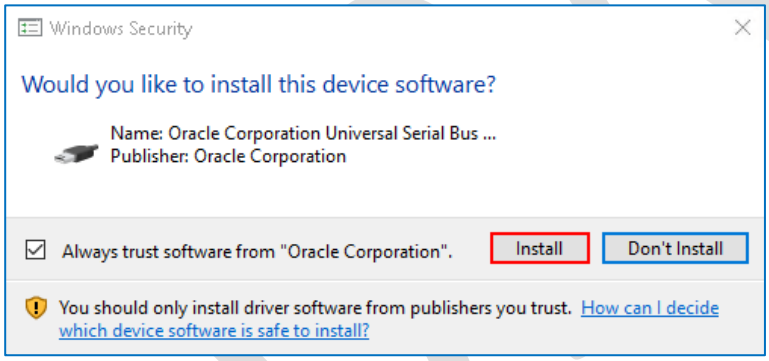
- Click **Next**



- Click **Next**



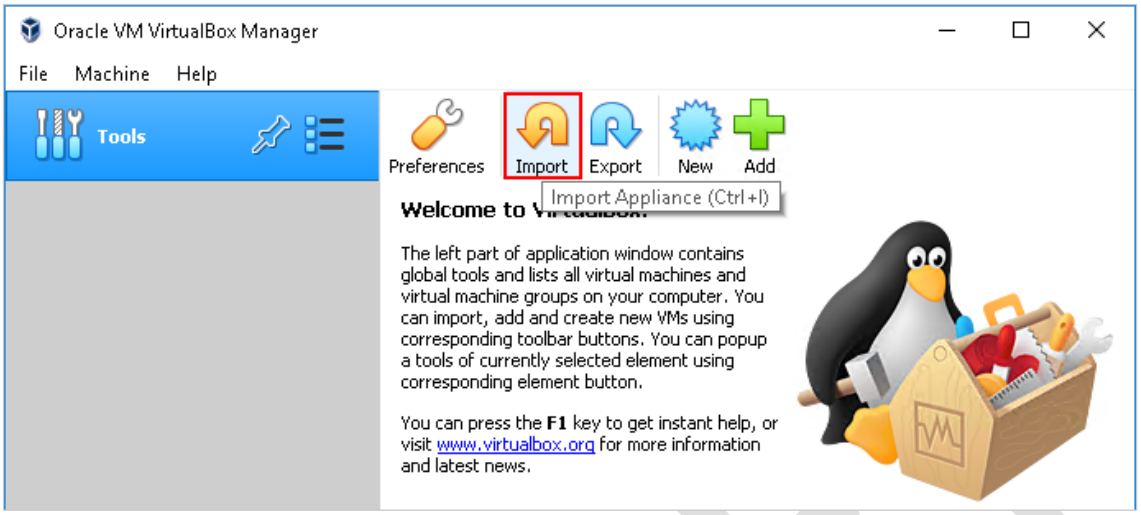
- Then click **Yes** to proceed with the installation then click **Install**
- When the Windows Security message pops-up, click **Install**



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

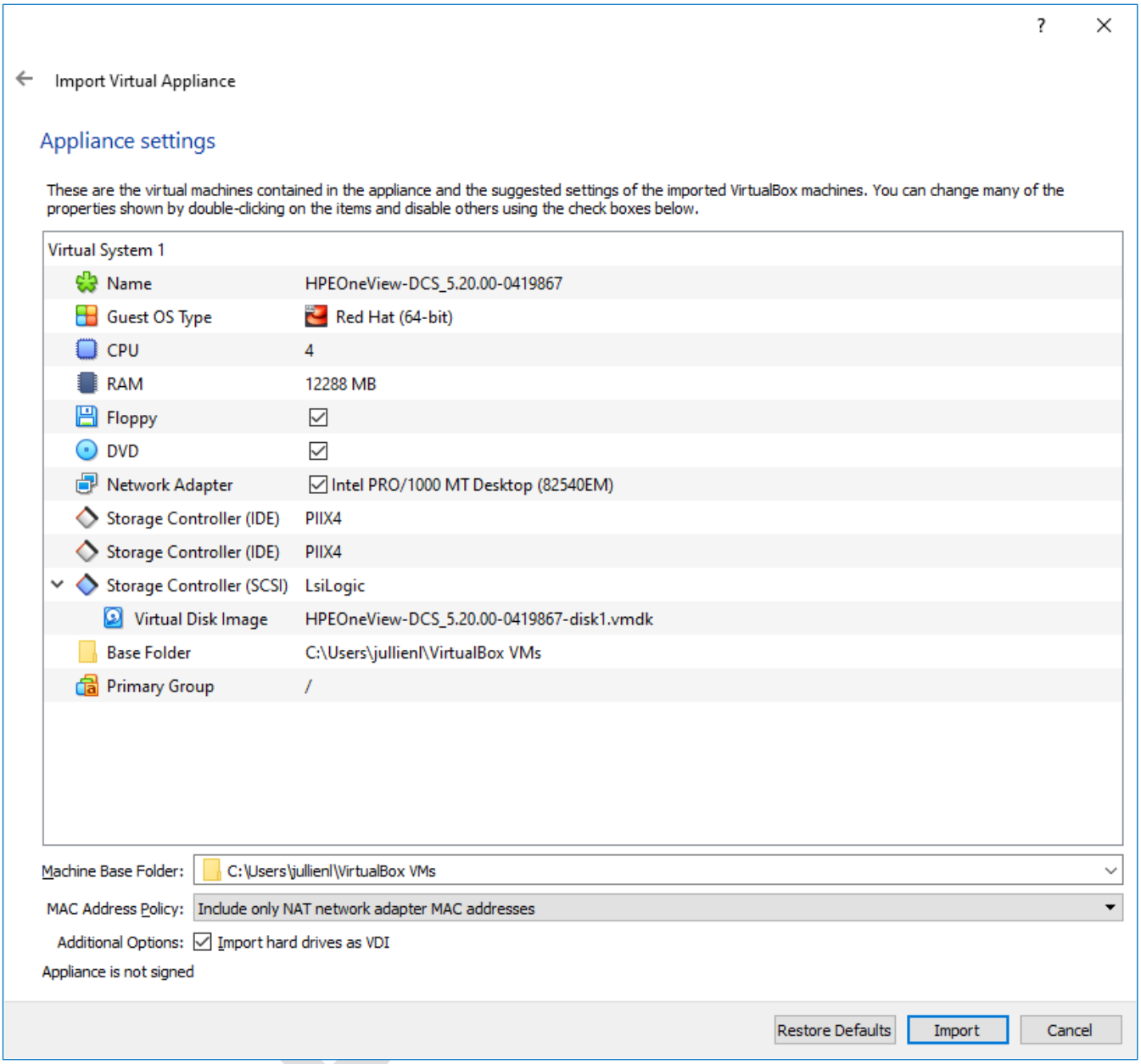
Importing and tuning the HPE OneView Demonstration appliance in VirtualBox

- In VirtualBox, select **Import**

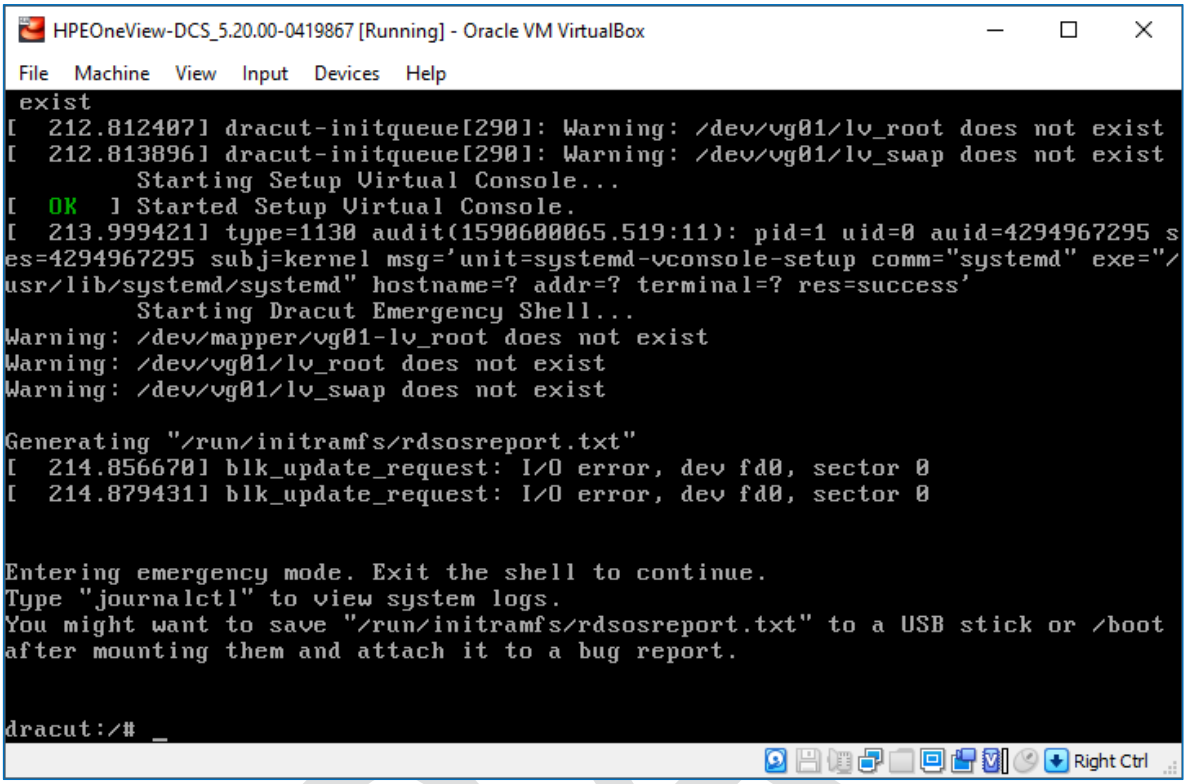


- Select the **HPE_OneView_DCS_5.20_Synergy_ESXi_Z7550-96879.ova** file

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

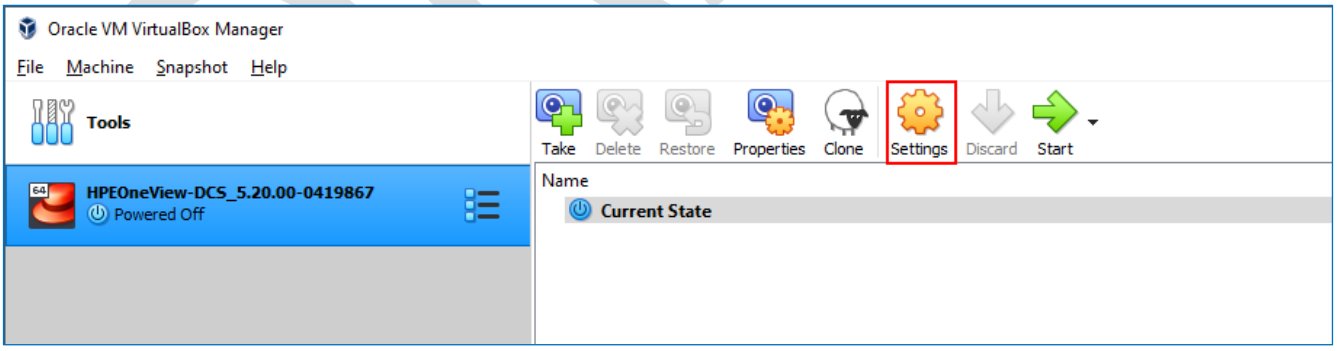


Note: If you start the VM in the current state, the VM is entering in emergency mode because the driver of the disk controller simulated by VirtualBox is not available.

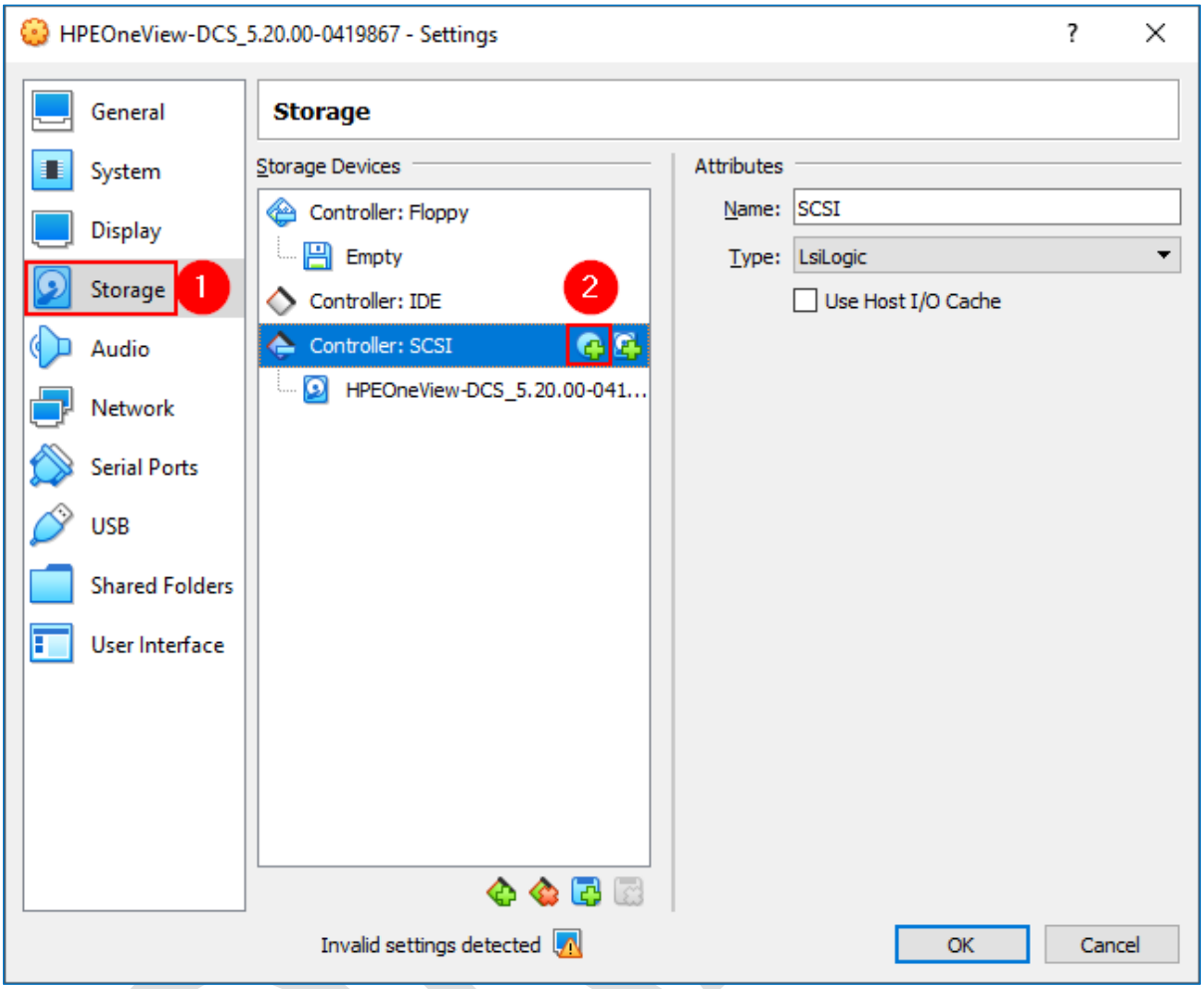


The solution to fix this issue is to inject the missing drivers from the downloaded CentOS 7.5 CD ISO.

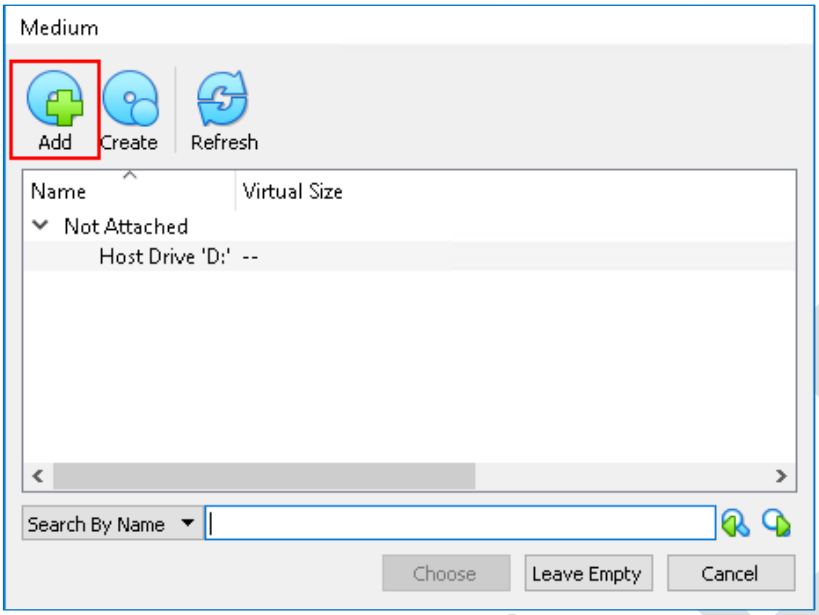
- Edit the VM Settings



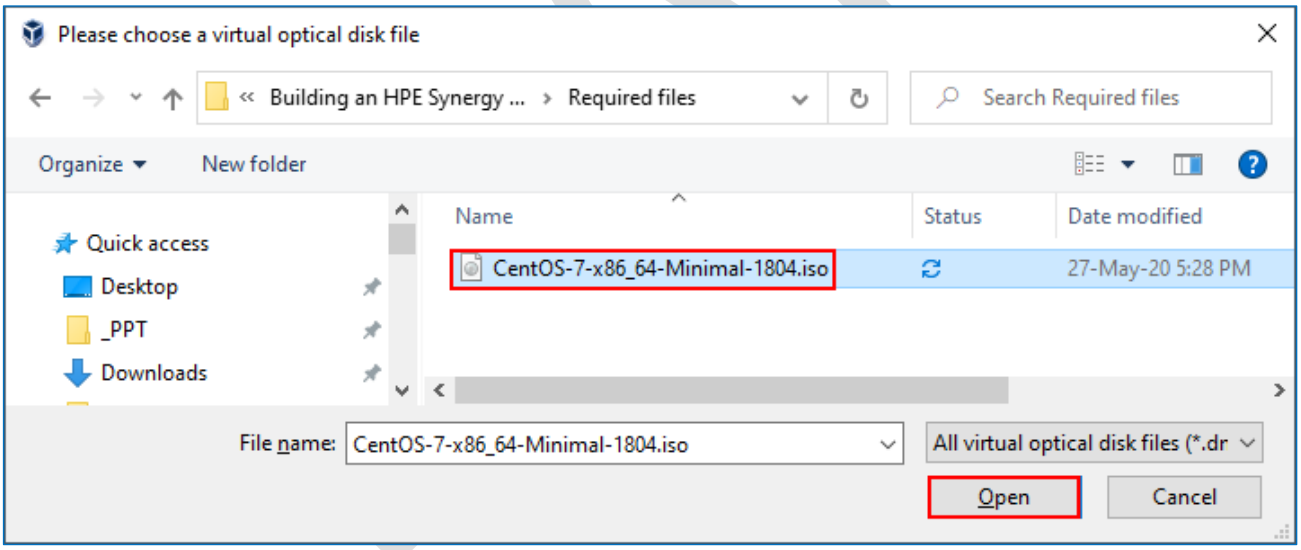
- Select **Storage** then click on **Adds optical drive** icon on the iSCSI Controller



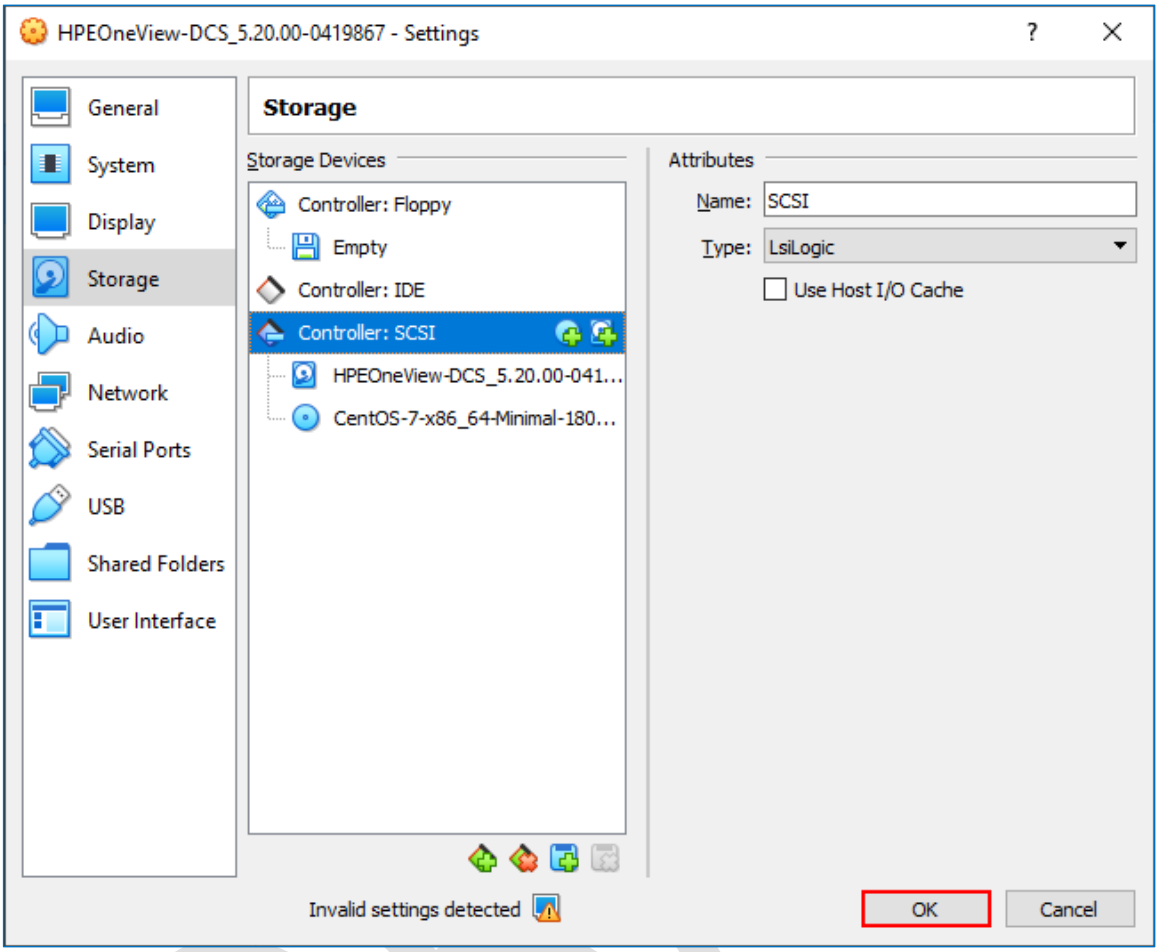
- Select **Add**



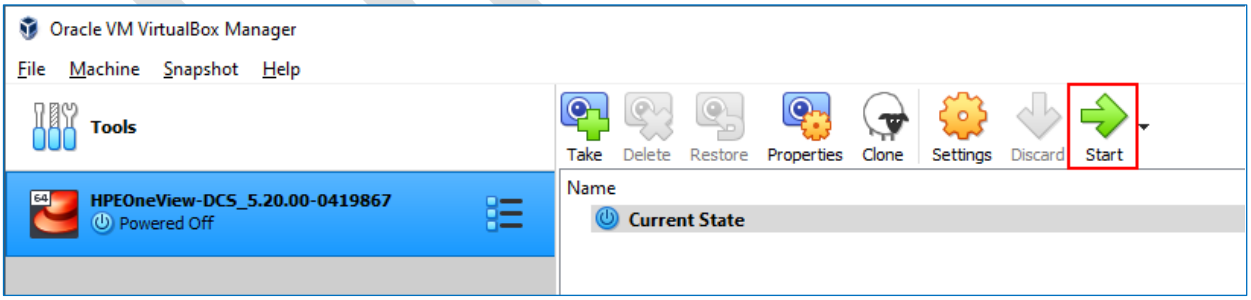
- Select the CentOS ISO image then click **Open**



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



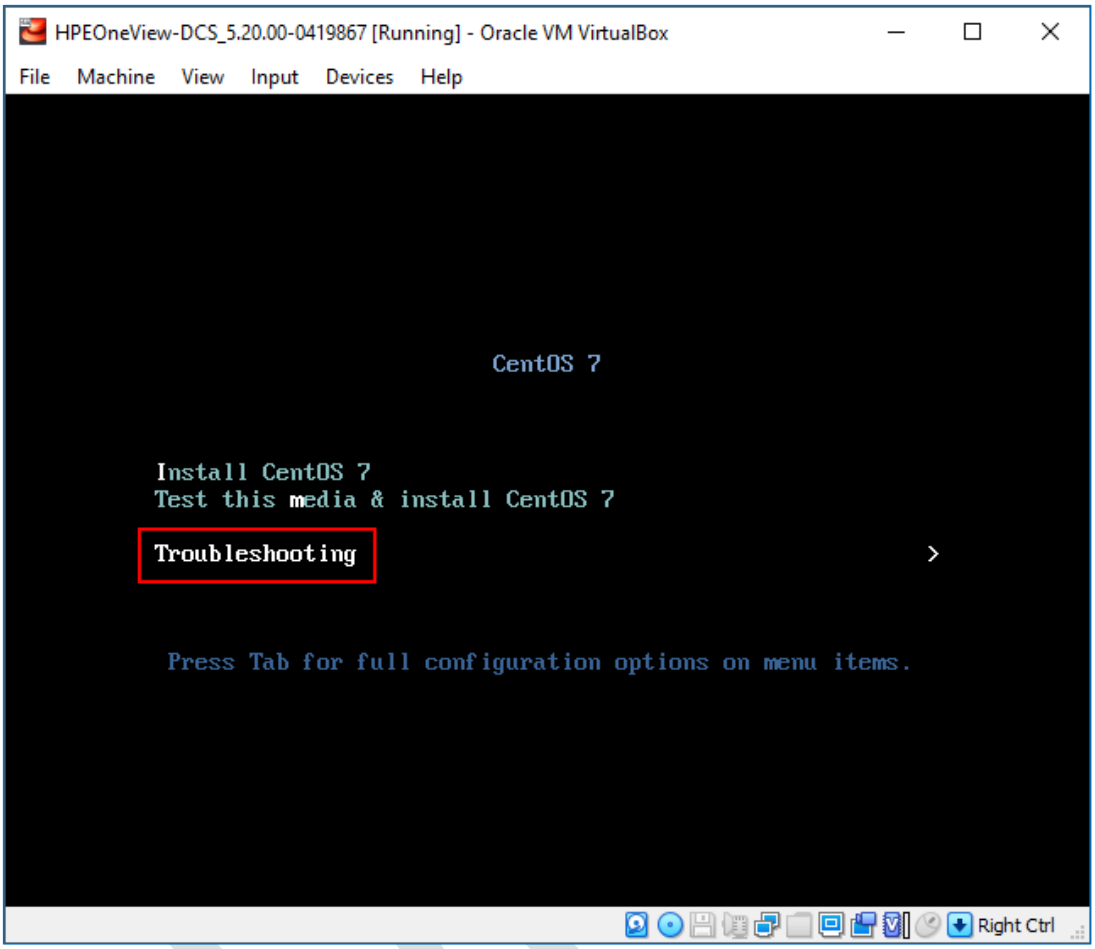
- Then start the VM by pressing **Start**



The CentOS starting menu should come up.

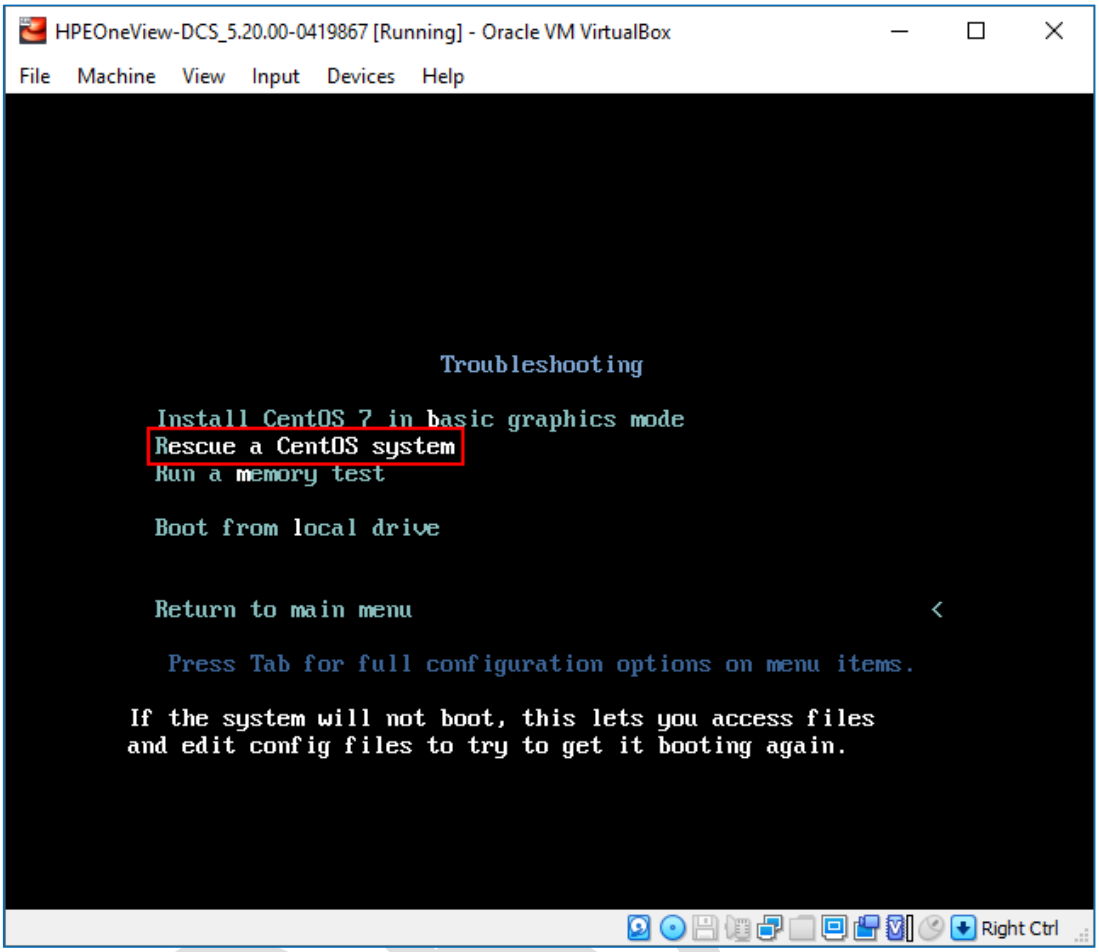
Note: If you are seeing some errors when you start the VM, go to the Troubleshooting section

- Select **Troubleshooting**

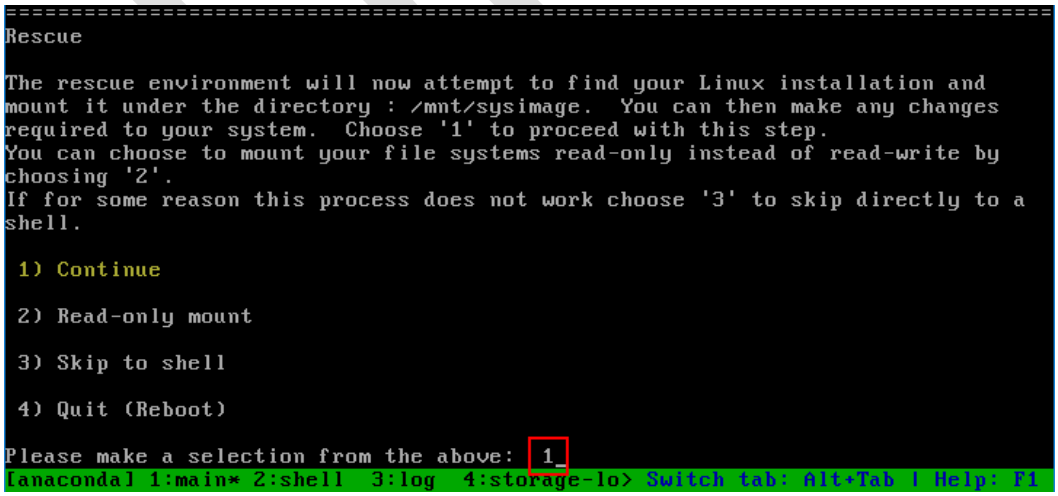


Note: To release the mouse from a VirtualBox console use the right Control key on your keyboard

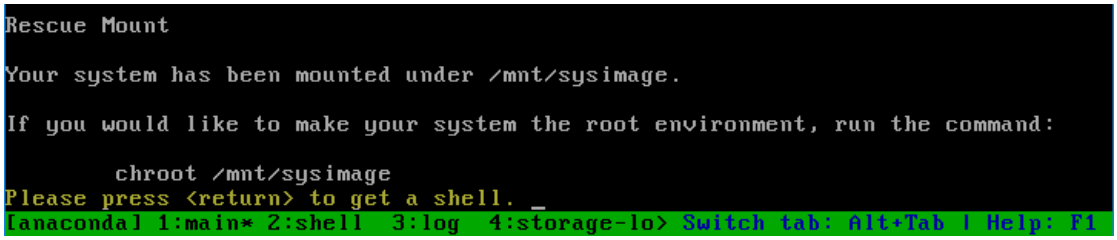
- Then select **Rescue a CentOS system**



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



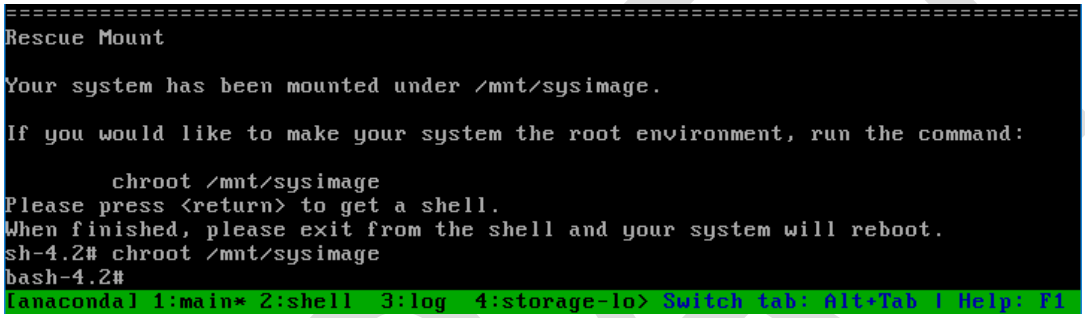
- Press **ENTER** to get a shell.



- 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



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

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

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

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

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

As part of the OneView 5.20 Security Hardening process, SELinux has been enabled. On some laptops while testing, we observed that SELinux gets in the way and the DCS does not load properly. Perform the following steps to relax the SELinux settings.

- Change to the SELinux Config folder:

```
cd /etc/selinux
```

- Open the config file for editing:

```
vi config
```

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

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are 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 exit command to unmount the appliance Image and return to the Rescue Mode shell.:

```
 exit
```

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

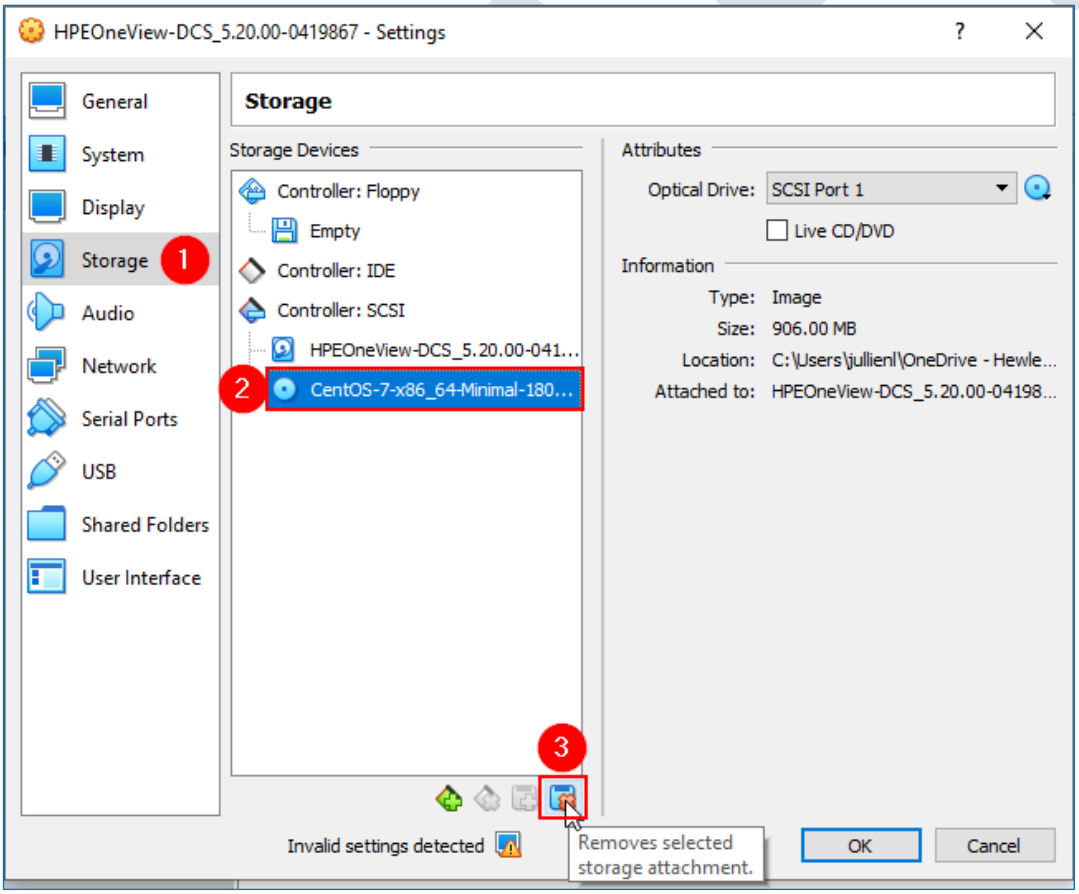
The command prompt should change from *bash* back to *sh*.

- Then shut down the VM:

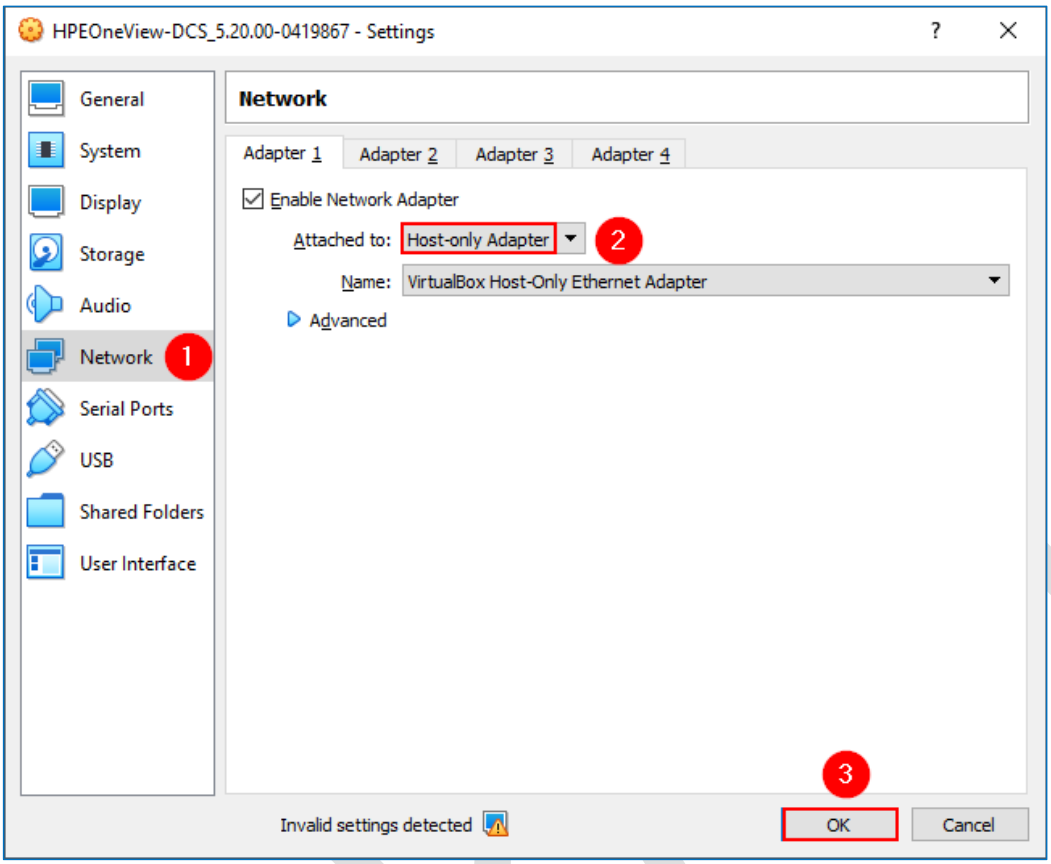
```
shutdown now
```

```
[  OK  ] Unmounted /mnt/sysimage/dev/shm.
[  OK  ] Unmounted /mnt/sysimage/dev/pts.
[  OK  ] Unmounted /mnt/sysimage/boot.
         Unmounting /mnt/sysimage/dev...
         Unmounting /mnt/sysimage/sys...
[  OK  ] Unmounted Temporary Directory.
[  OK  ] Unmounted /mnt/sysimage/sys.
[  OK  ] Unmounted /mnt/sysimage/dev.
         Unmounting /mnt/sysimage...
[  OK  ] Stopped target Swap.
         Deactivating swap /dev/dm-9...
[  OK  ] Deactivated swap /dev/vg01/lv_swap.
[  OK  ] Deactivated swap /dev/mapper/vg01-lv_swap.
[  OK  ] Deactivated swap /dev/disk/by-uuid/...030-c193-4454-a09a-43ca1b824832.
[  OK  ] Deactivated swap /dev/disk/by-id/dm...LGNt6qti9MVIaJSQ9Mm0de0TjV8hxJi.
[  OK  ] Deactivated swap /dev/disk/by-id/dm-name-vg01-lv_swap.
[  OK  ] Deactivated swap /dev/dm-9.
[  OK  ] Unmounted /mnt/sysimage.
[  OK  ] Reached target Unmount All Filesystems.
[  OK  ] Stopped target Local File Systems (Pre).
[  OK  ] Stopped Remount Root and Kernel File Systems.
         Stopping Remount Root and Kernel File Systems...
[  OK  ] Stopped Create Static Device Nodes in /dev.
         Stopping Create Static Device Nodes in /dev...
_
```

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



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

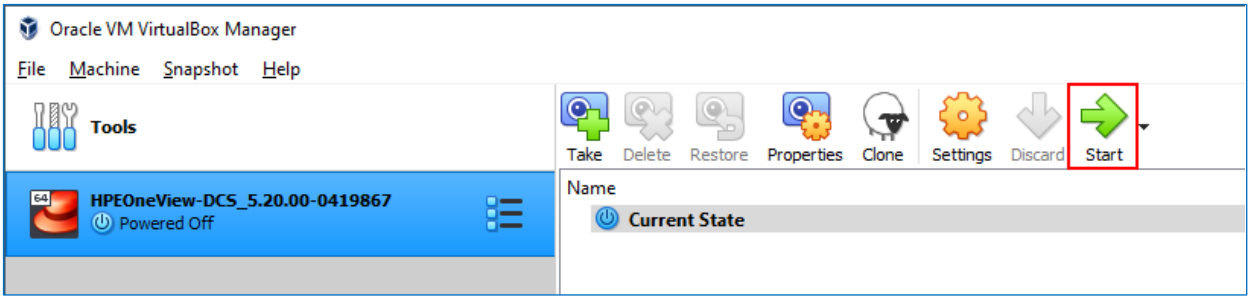


- Click **OK** to save changes.

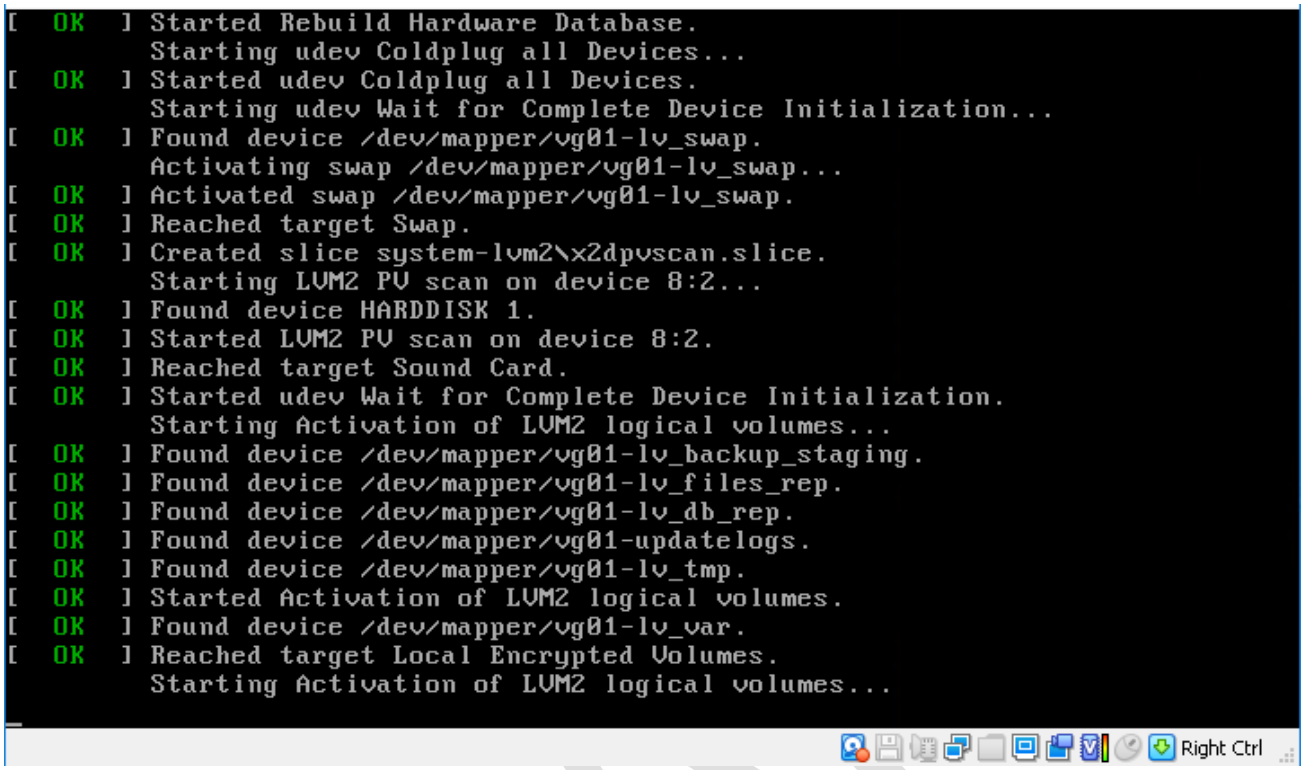
- Shutdown all applications running on your computer before initiating the startup process. This relieves memory/CPU/disk pressure on the VM while booting.

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

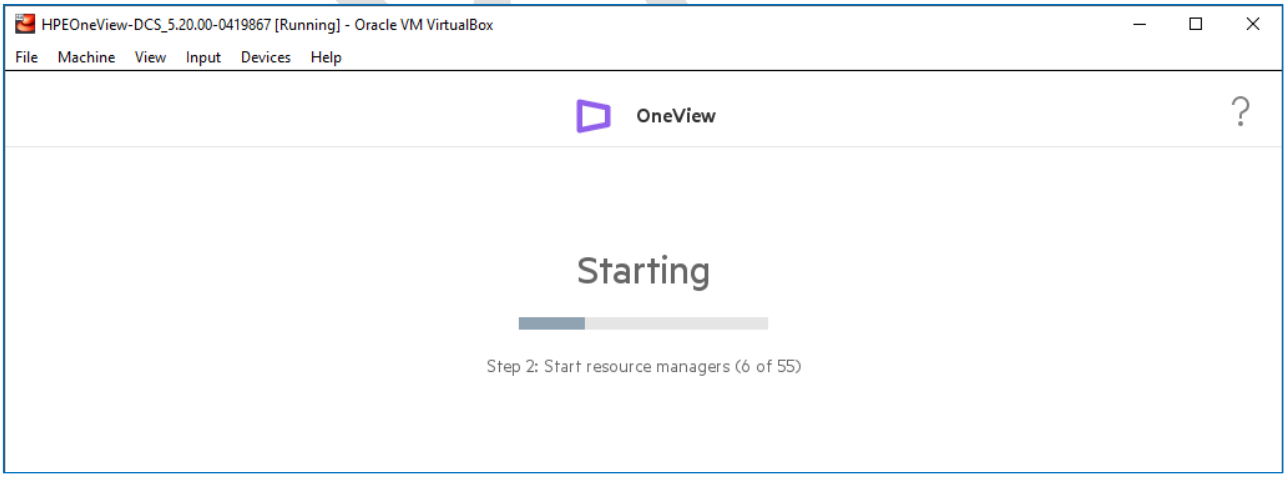
- Click on **Start** to power on the HPE OneView demonstration appliance VM.



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



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



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

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

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

This concludes Chapter-1

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

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

Installing Windows Subsystem for Linux (WSL)

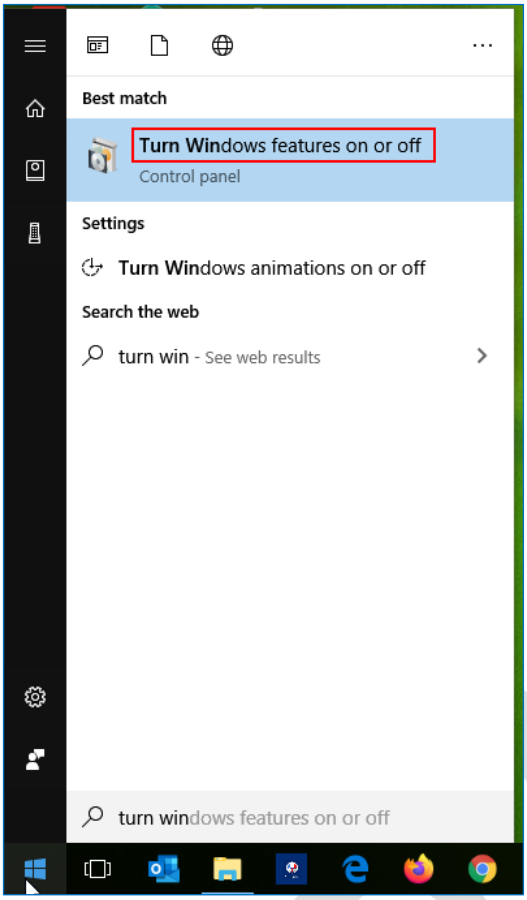
Windows Subsystem for Linux (WSL) is an optional feature on Windows 10 that creates a lightweight environment that allows you to install and run supported versions of Linux (such as Ubuntu, OpenSUSE, Debian, etc.) without the complexity and overhead of a virtual machine. It is light, fast and easy to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

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

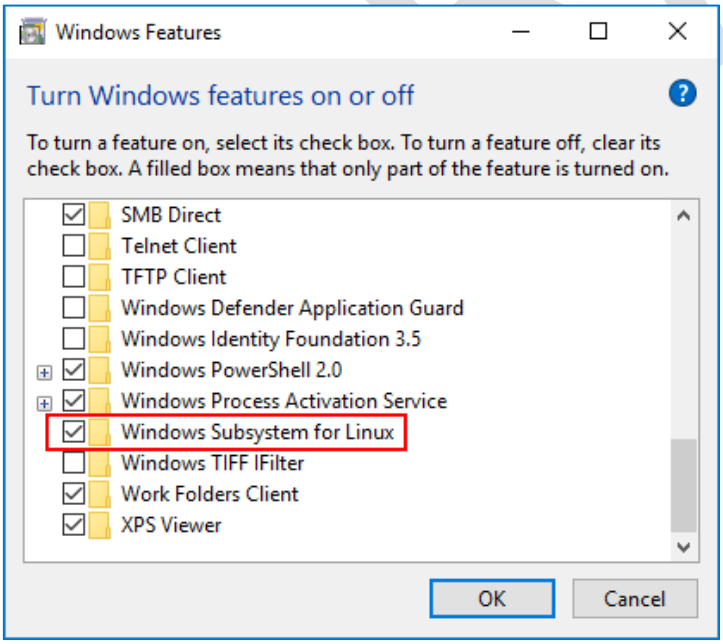
For our PC demonstration environment, we are going to use WSL (version 1) to run Python, Go, Ansible and Terraform.

> **Note**: It is recommended to use a recent version of WSL (Windows 10, April 2018 update, version 1803) to get better performance, more compatibility with Linux-native applications and VS Code extensions. (You can run *winver* to find the Windows version you are running).
>
> **Note**: The old version of WSL in Windows version 1709 can also be used for this lab.
>
> **Note**: WSL2 is a new version of the architecture in WSL that is increasing file system performance and adding full system call compatibility, see https://docs.microsoft.com/en-us/windows/wsl/wsl2-index.
> WSL 2 is only available in Windows 10, Version 2004, Build 19041 or higher.
> We do not recommend using WSL2 as there are many known compatibility issues with Oracle VM VirtualBox as unlike WSL, WSL2 is using Hyper-V and for now, Hyper-V does not play nicely with any other virtualization platform.

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



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



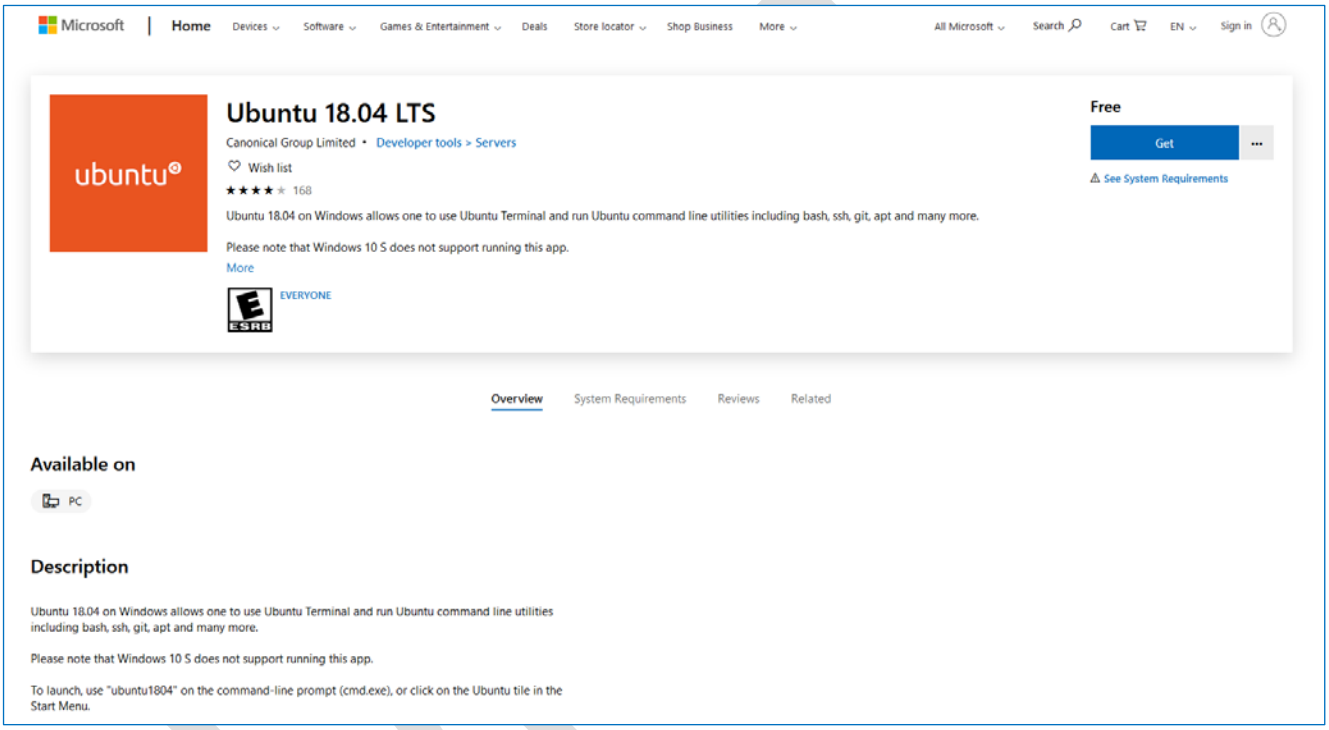
- Once turned on, follow the instructions, and do any restarts if you have to.

Installing Ubuntu Windows Subsystem for Linux

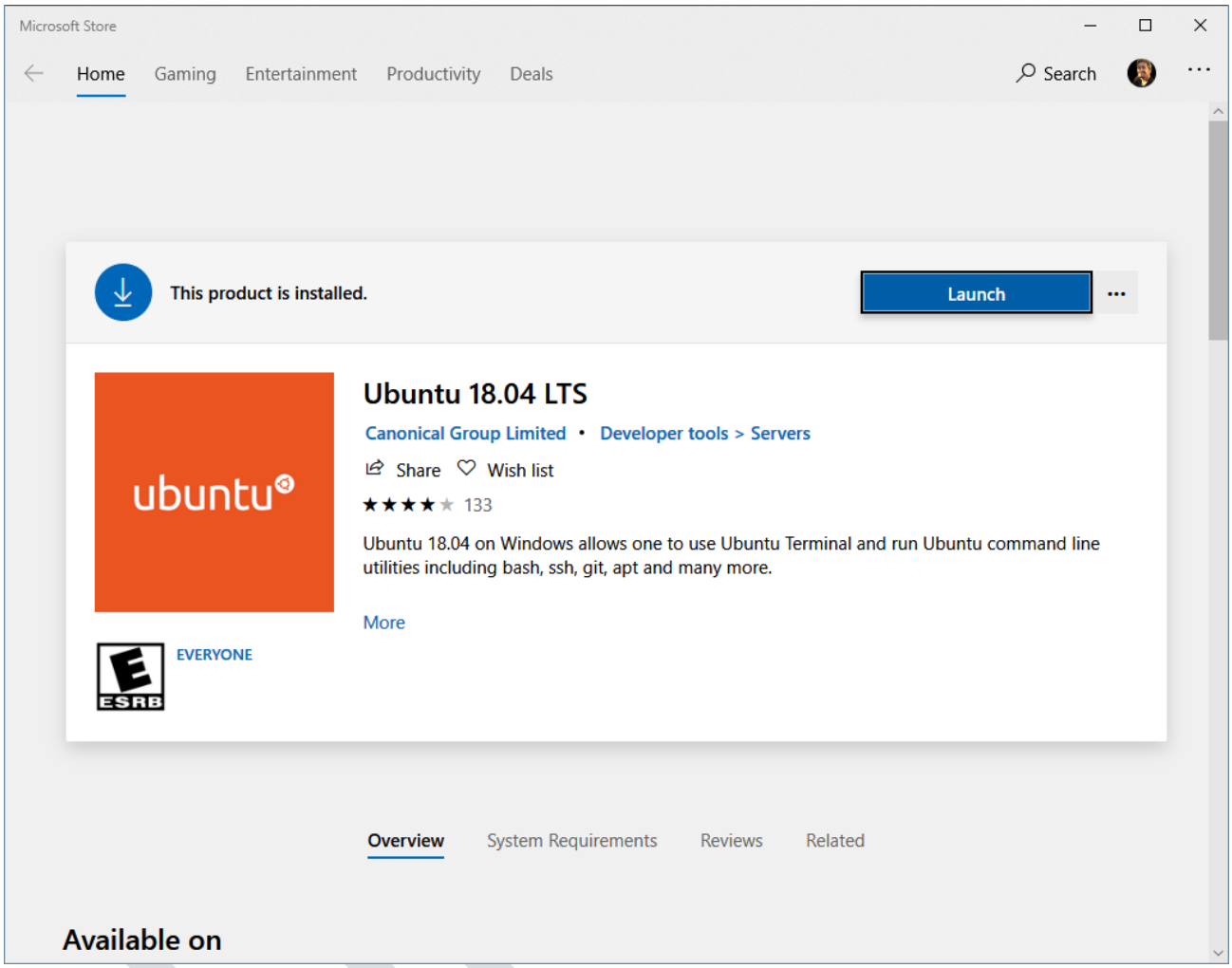
Next, we need to install a Linux distribution, we are going to use Ubuntu since that is one of the most popular.

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

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



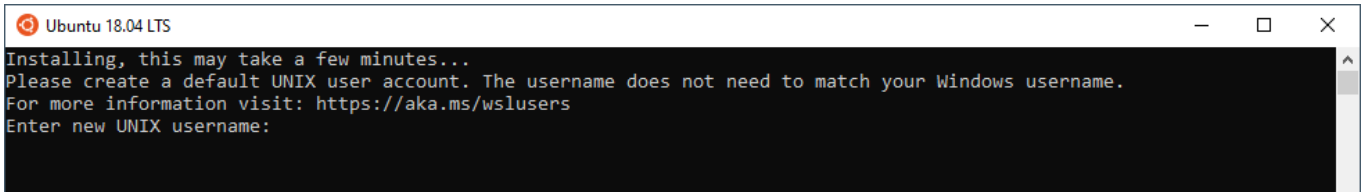
- Once Ubuntu has been downloaded and installed, click the **Launch** button in the Microsoft Store app, or launch Ubuntu from the Start menu or type `wsl` in a windows command prompt.



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

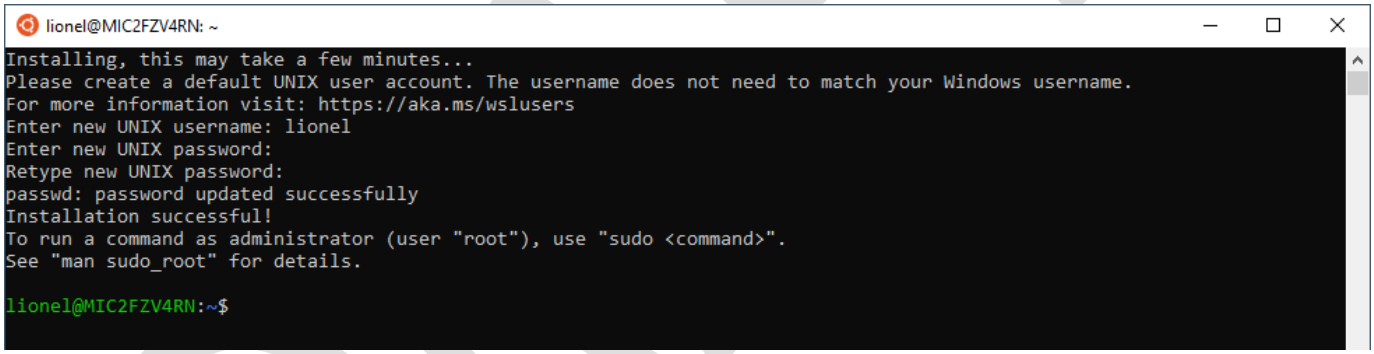
Note that the installation may take longer time depending on the performance of your PC's storage devices. This initial installation phase is only required when a distro is clean-installed - all future launches should take less than a second.

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



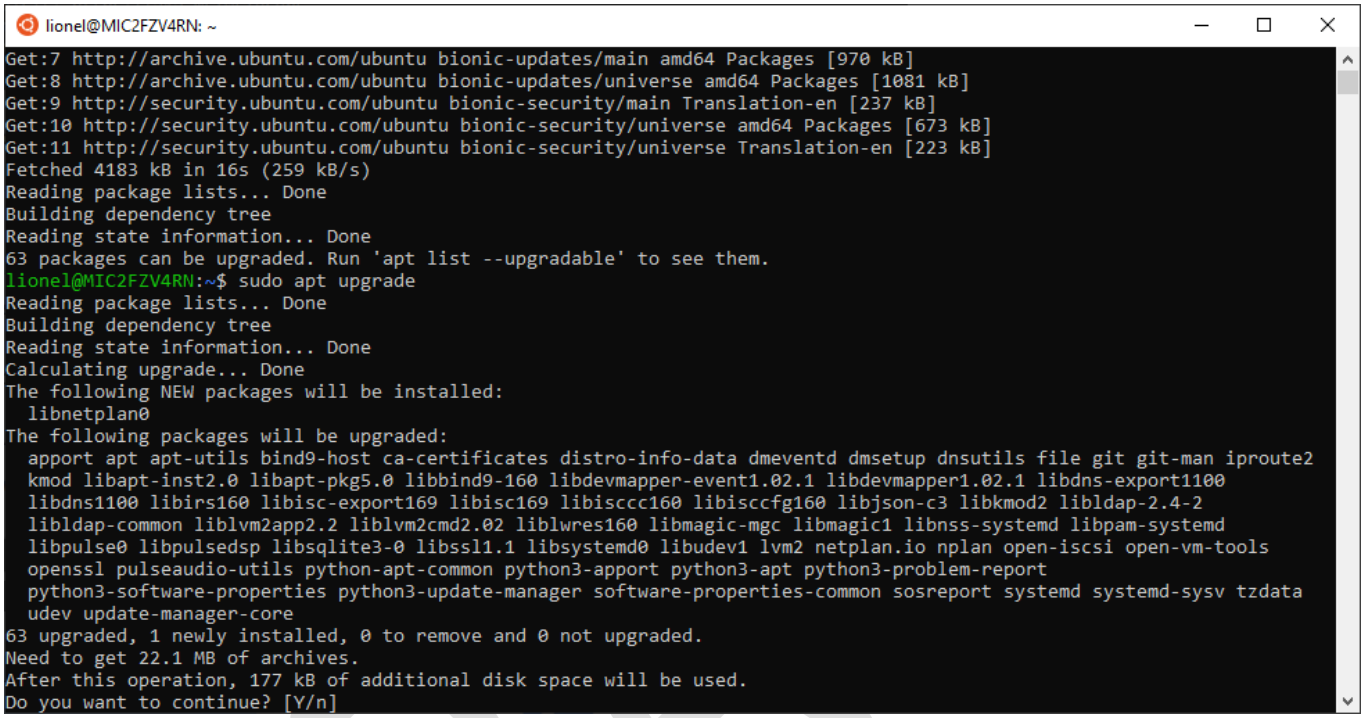
- This user account is for the normal non-admin user that you will be logged-in as by default when launching Ubuntu.

> **Note**: You can choose any *username* and *password* you wish - they have no bearing on your Windows username.
>
> **Note**: When you open a new distro instance, you won't be prompted for your password, but if you elevate a process using *sudo*, you will need to enter your password, so make sure you choose a password you can easily remember!



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

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

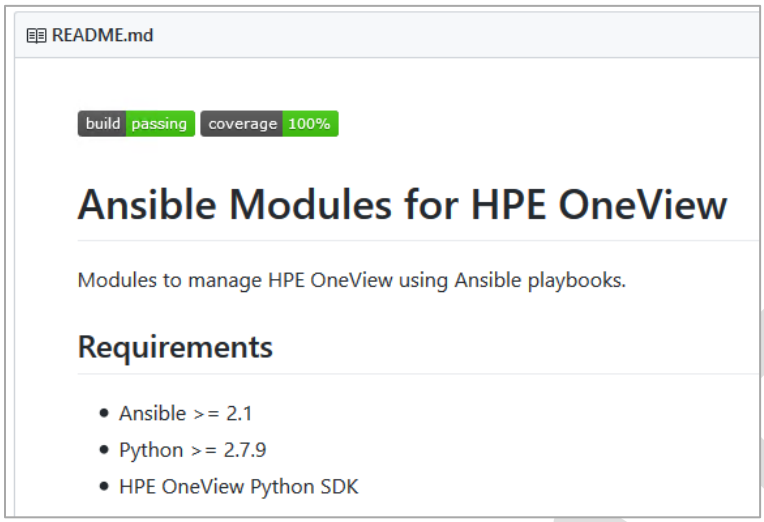


- When prompted, type **Yes** to install all the new packages

> **Note**: Closing the WSL window does not stop Ubuntu. If you need to restart WSL Ubuntu, you need to type in a Windows command prompt:
>
> ```
> wslconfig /l
> wslconfig /t Ubuntu-xx.xx
> ```
>
> This command terminates the Ubuntu distribution. Once it is closed, you would need to restart Ubuntu WSL.
>
> 
>

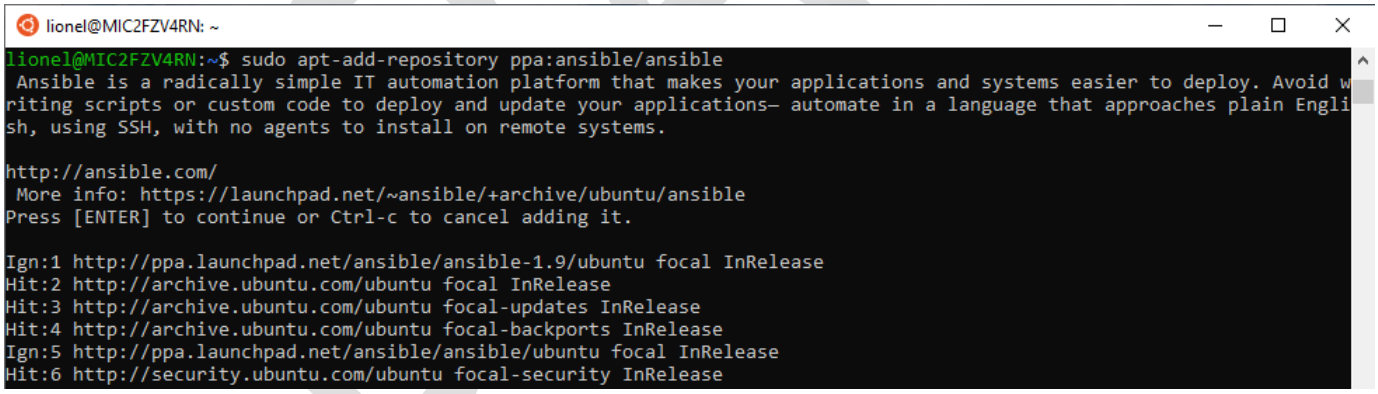
Installing Ansible on Ubuntu WSL

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



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

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

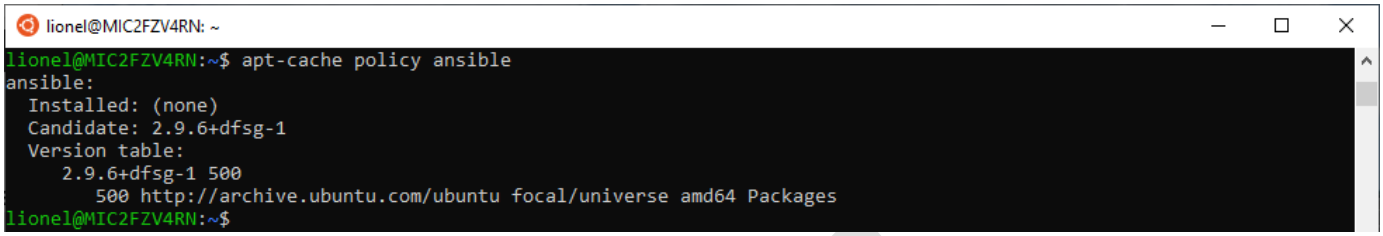


- Press **Enter** when prompted.

> **Note**: if you are behind a corporate proxy, you need to set the proxy environment variables:
>
> ```
> export http_proxy=http://proxy.myproxy.com:port
> export https_proxy=https://proxy.myproxy.com:port
> ```
>
> 
>

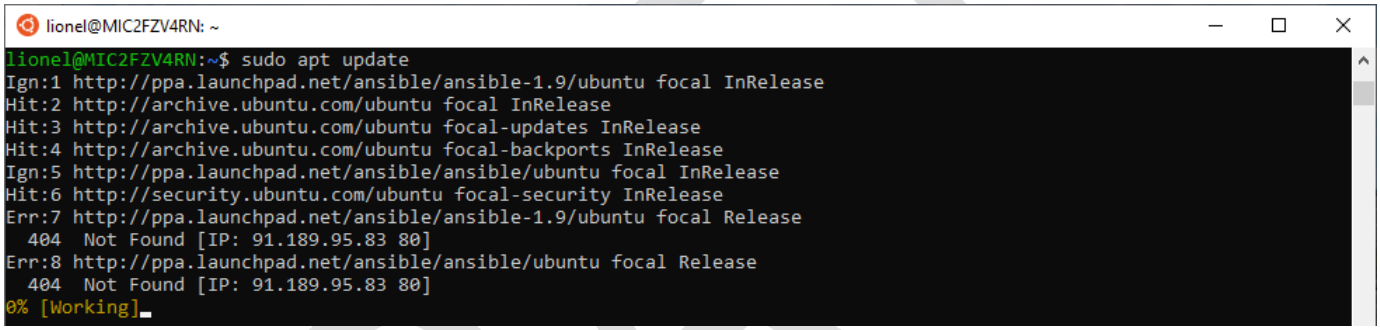
- To check all available packages in the repository, enter:

```
apt-cache policy ansible
```



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

```
sudo apt update
```

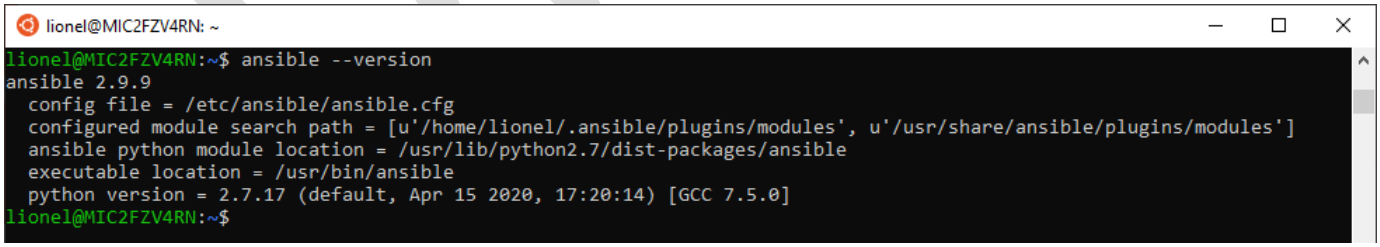


- Then we can install Ansible with:

```
sudo apt install ansible
```

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

```
ansible --version
```

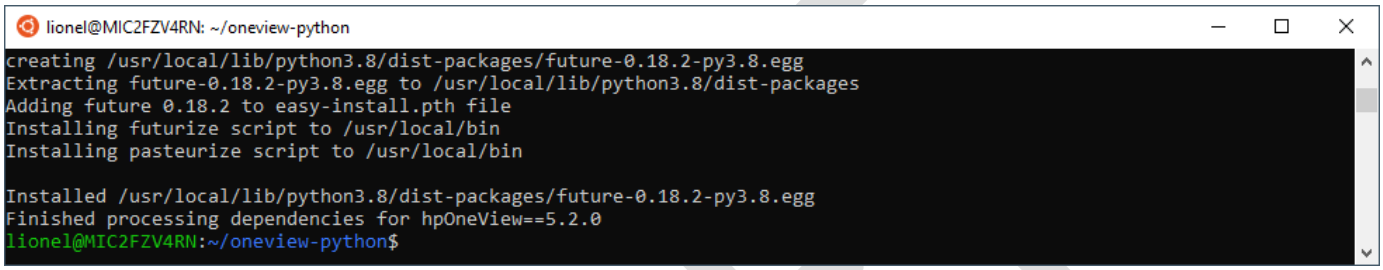


Installing HPE OneView Python SDK on Ubuntu WSL

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

- Type:

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



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

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

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

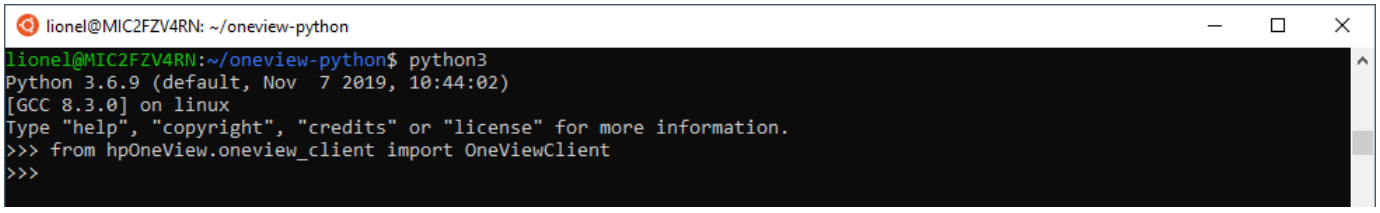
> **Important notice**: Do not to use the *python-hpOneView* repository from https://github.com/HewlettPackard/python-hpOneView ! This is the repository to support legacy SDKs
>
> **Note**: if you are behind a corporate proxy, you need to set a Git proxy:
>
> ```
> git config --global http.proxy http://proxy.myproxy.com:port/
> git config --global https.proxy http://proxy.myproxy.com:port/
> ```

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

```
python3
```

- Then import the HPOneView module with:

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



- 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))
```

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

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

Installing Ansible Modules for HPE OneView on Ubuntu WSL

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

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

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

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

```
cd ~
```

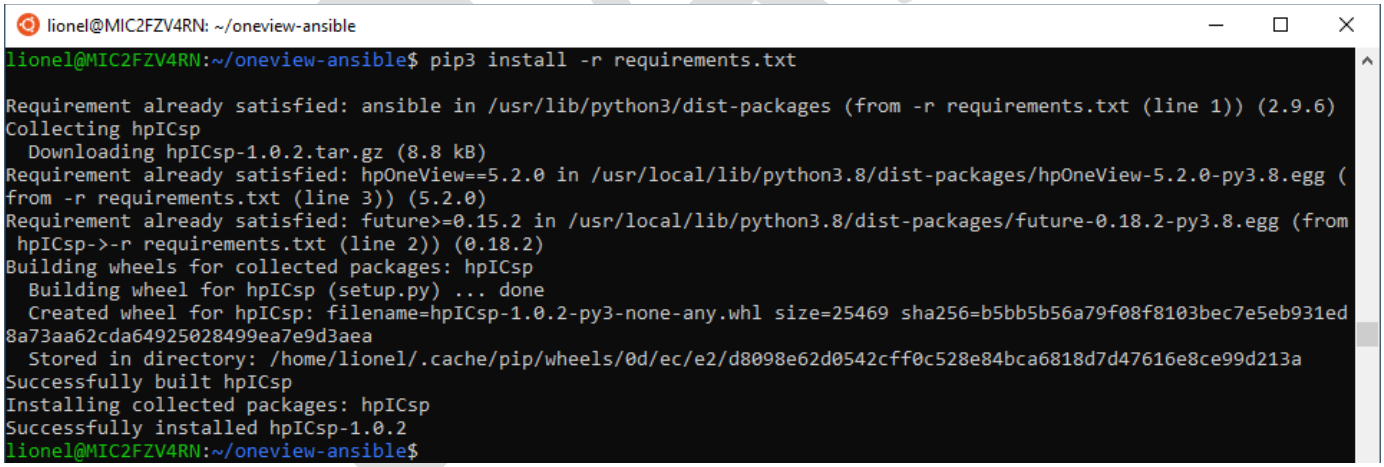
- Then use *git clone* to clone the Ansible Modules for HPE OneView repository:

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

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

- Then install the dependency packages:

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



> **Note**: if you are behind a corporate proxy, you need to define a proxy:
>
> ```
> pip3 install -r requirements.txt --proxy http://proxy.server.com:port/
> ```

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

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

- We currently have only one Master branch:

```
git branch
```

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

- To navigate to the fix branch, enter:

```
git checkout bug_fix/create_profiles_in_parallel
```

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

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

```
git branch
```

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

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

```
git checkout master
```

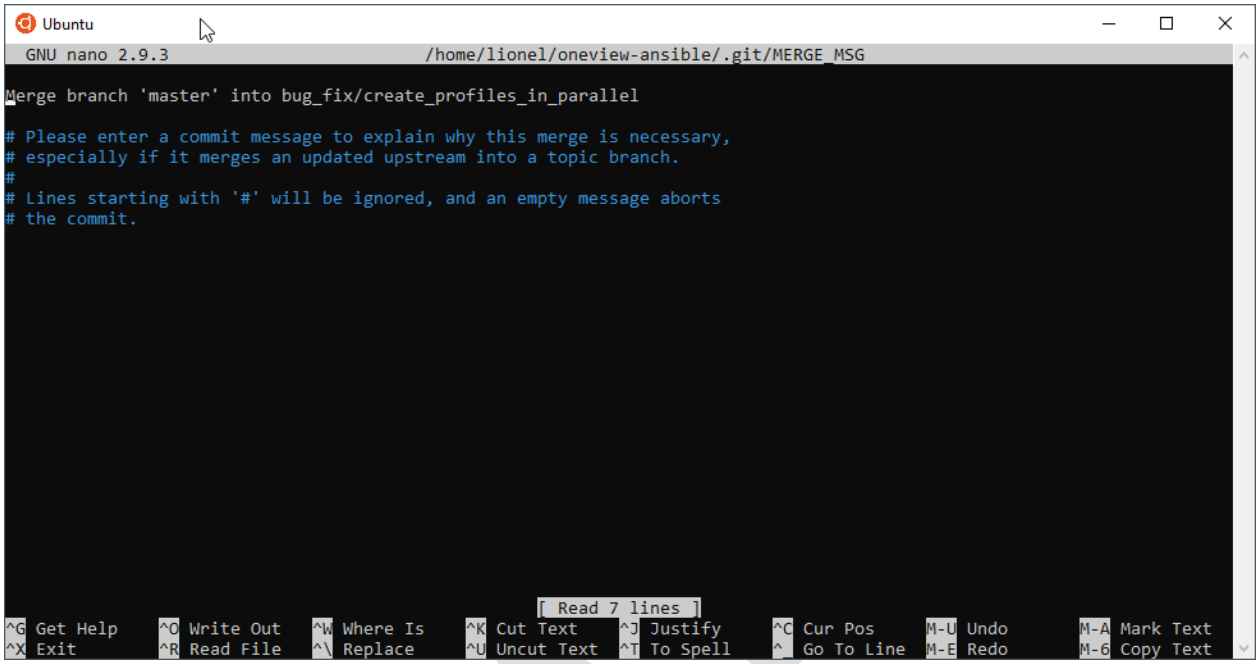
Then to merge the bug_fix features into the *master* branch, run:

```
git merge bug_fix/create_profiles_in_parallel
```

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

> **Note**: If you get an '*empty ident name not allowed*' error when running the *git merge* command, then you need to set a git account, it could be a fake one like:
>
> ```
> git config --global user.name "hpe"
> git config --global user.email "hpe@synergy.lab"
> ```

- The following nano editor should open:



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

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

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

- If you type now

```
git checkout master
```

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

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

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

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

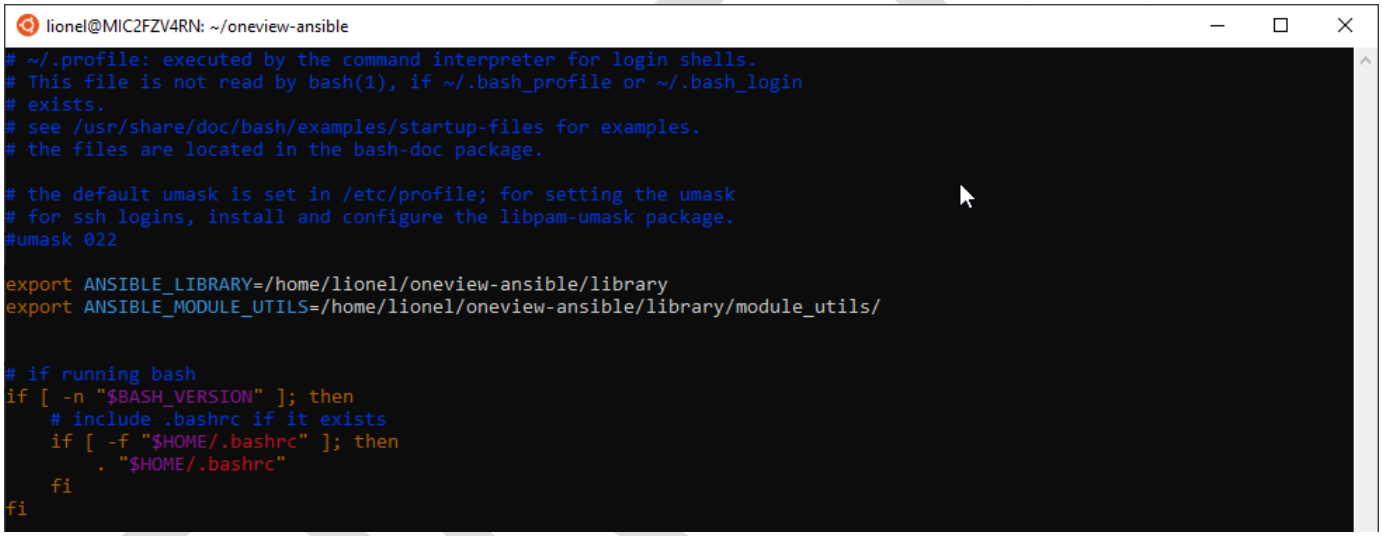
- Type:

```
vi ~/.profile
```

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

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

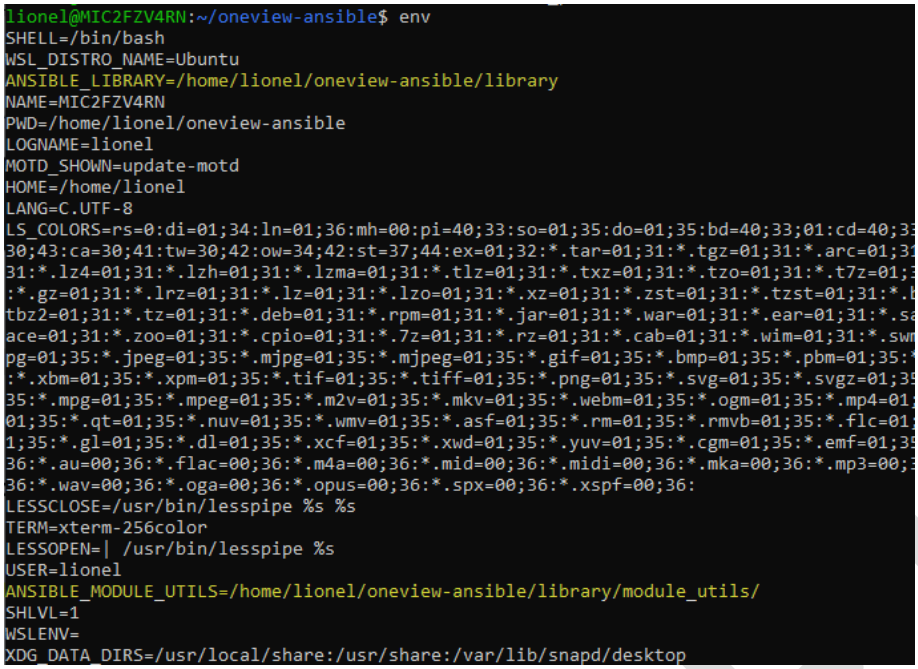
Note: Make sure you modify *<username>* with the username you have defined.



- To immediately apply the new environment variables, type:

```
source ~/.profile
```

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



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

```
cd ~
```

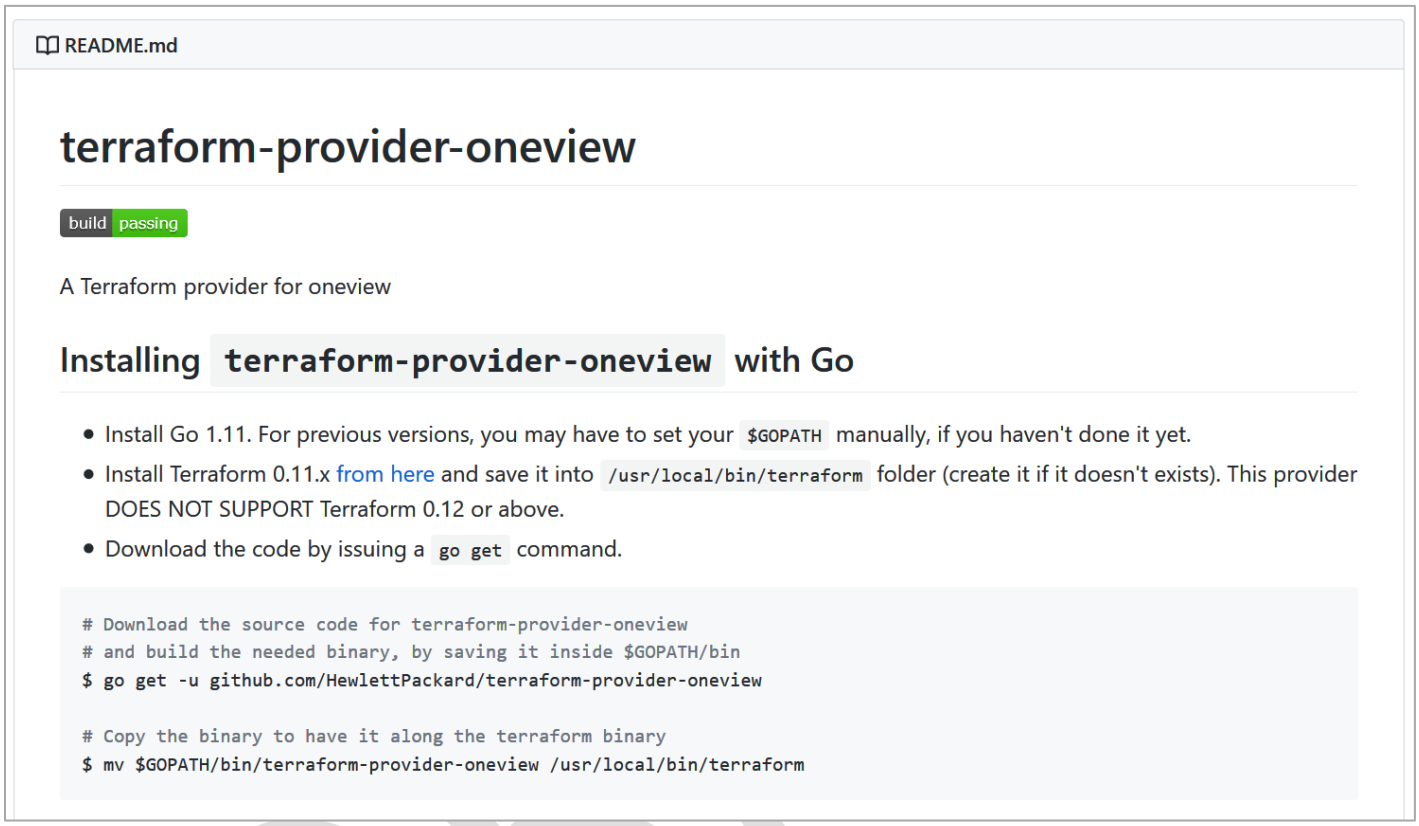
- Then create a **oneview_config.json** file with our appliance information:

```
cat > oneview_config.json << "EOF"
{
    "ip": "192.168.56.101",
    "api_version": 1600,
    "credentials": {
        "userName": "administrator",
        "authLoginDomain": "",
        "password": "password"
    }
}
EOF
```

> **Note**: *192.168.56.101* is the IP address we will configure later for the DCS appliance taken from the "Host-only" VirtualBox subnet (192.168.56.0/24).

Installing Terraform on Ubuntu WSL

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



README.md

terraform-provider-oneview

build passing

A Terraform provider for oneview

Installing `terraform-provider-oneview` with Go

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

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

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

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

- To download Terraform 0.11.14. enter:

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

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

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

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

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

- To verify Terraform is installed, run:

```
terraform -v
```

Installing Go on Ubuntu WSL

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

- From your home folder, enter:

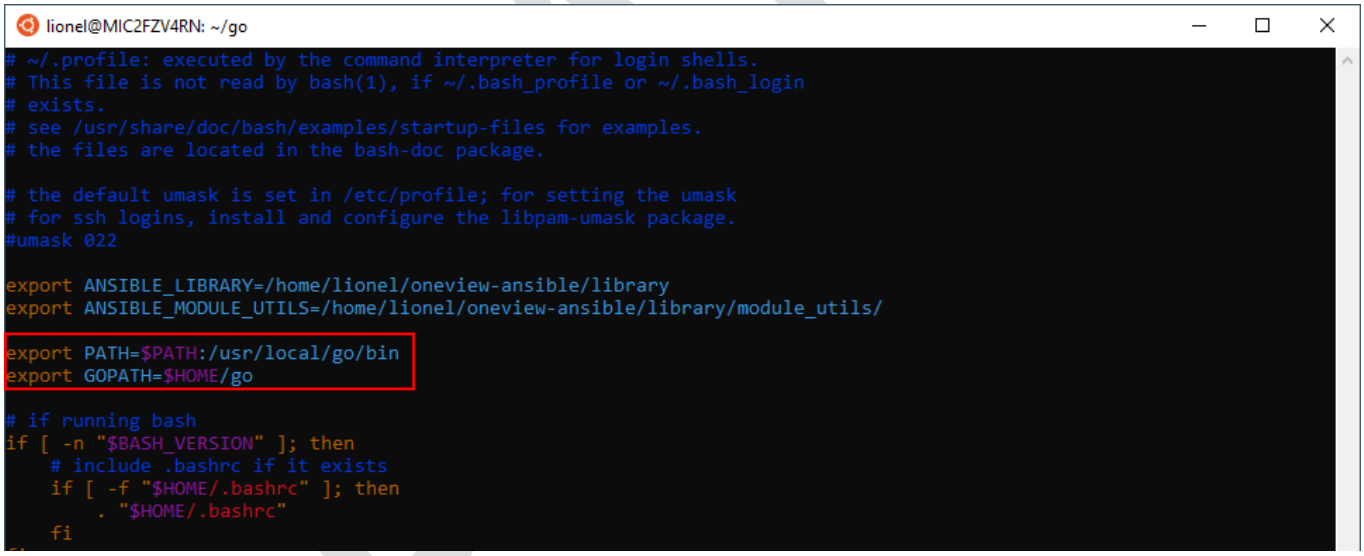
```
wget https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz
sudo tar -C /usr/local -xzf go1.14.2.linux-amd64.tar.gz
```

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

```
vi ~/.profile
```

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

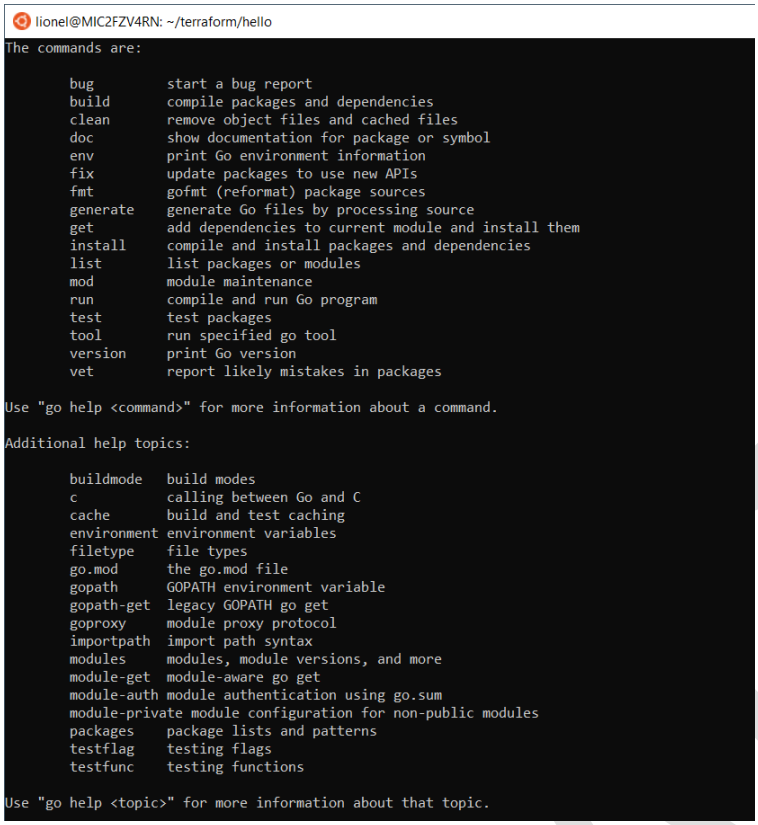
```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=$HOME/go
```



- To immediately apply the new environment variables, type:

```
source ~/.profile
```

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



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

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

  package main

  import "fmt"

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

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

```
go run hello.go
```

  You should see the console displaying *Hello, World*

Installing the Terraform provider for HPE OneView

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

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

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

This command performs the following actions:

1. Builds the needed binary and save it in `/home/<username>/go/bin` (= `$GOPATH/bin`)
2. Copies all source files in `home/<username>/go/src` which ends up being: `/home/<username>/go/src/github.com/HewlettPackard/terraform-provider-oneview/`

- The next step is to copy the generated binary into `/usr/local/bin` to have it along with the terraform binary

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

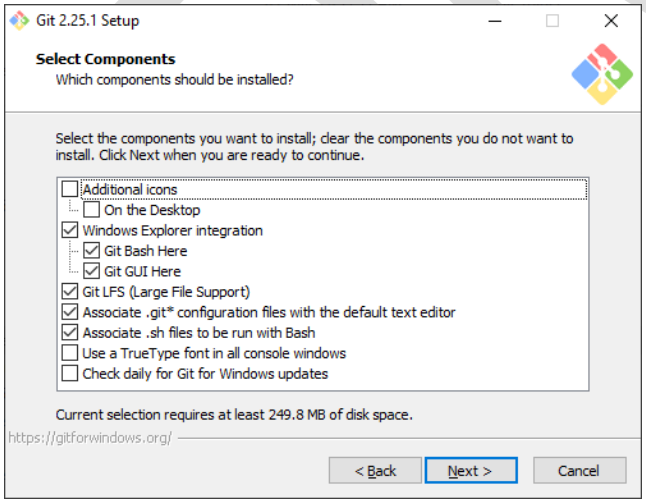
Installing Git for VS Code source control

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

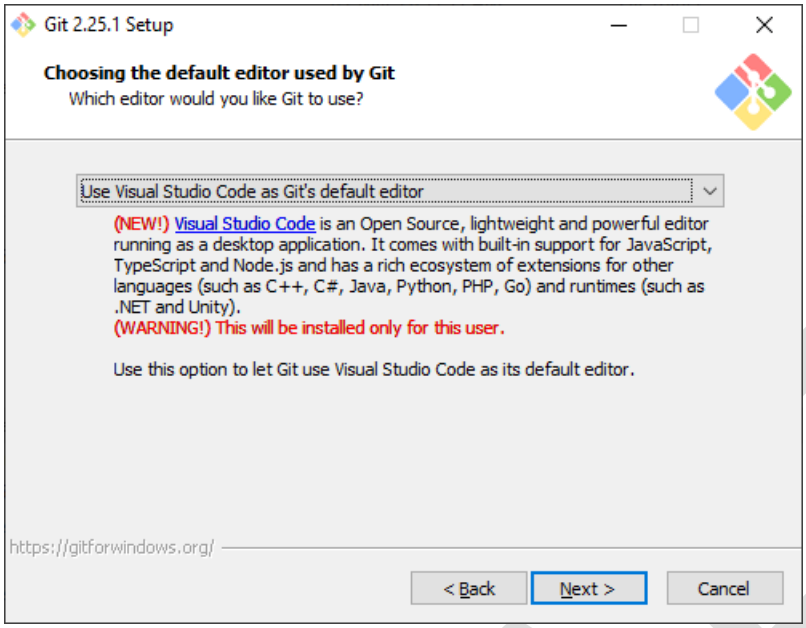
We are going to use Git in the PowerShell project to clone GitHub repositories in our VS Code workspace but also to make sure we always have the latest content that is regularly published by the developers. For the Linux/Ansible project, we will use the Git in Ubuntu WSL.

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

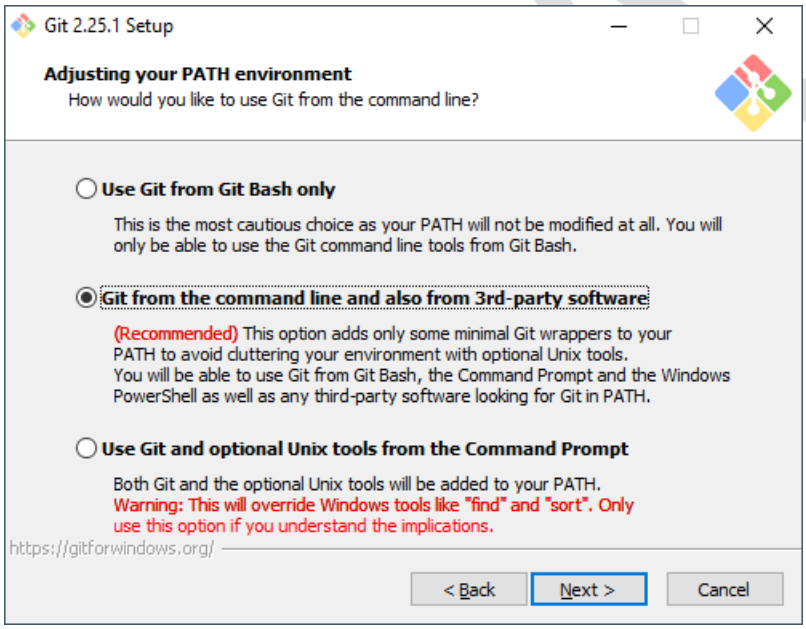
- Keep the proposed default components selected



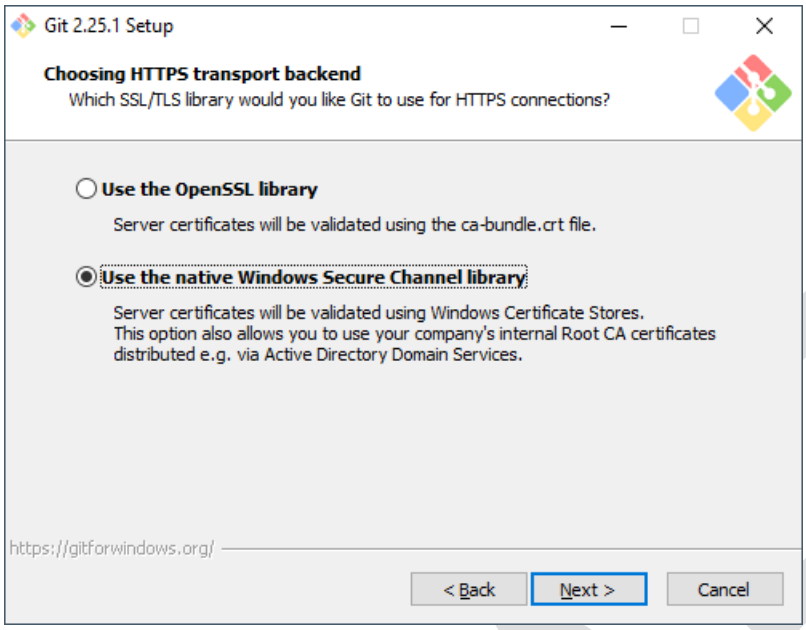
- Choose Visual Studio Code as Git's default editor option



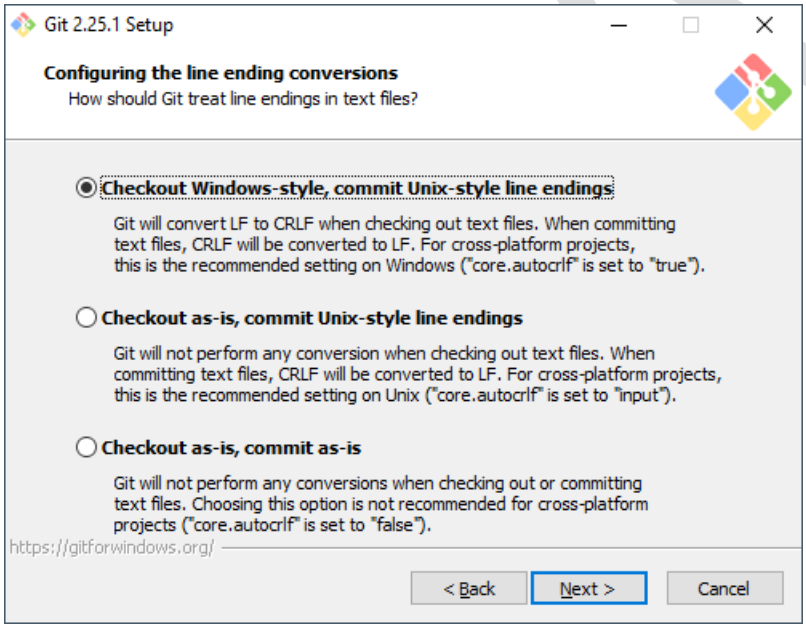
- Keep the recommended PATH environment option selected



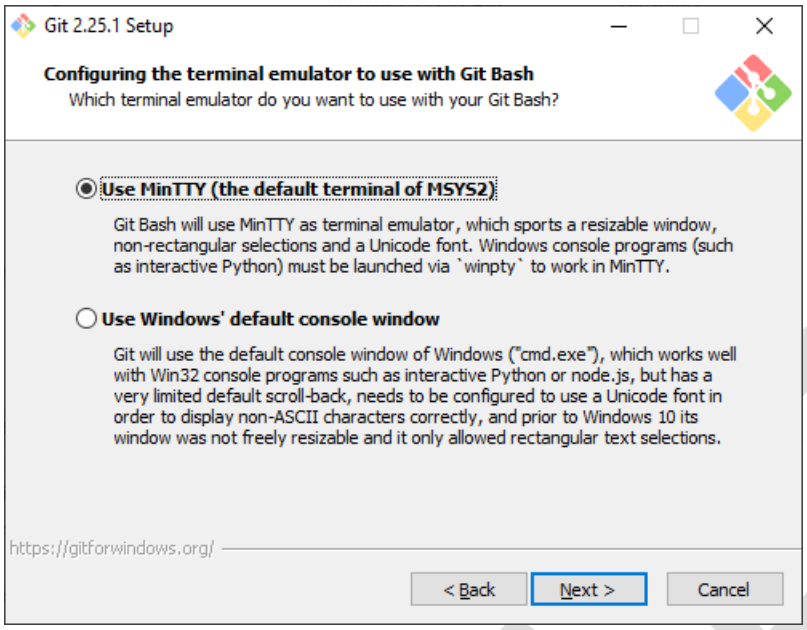
- Use the proper HTTPS transport backend option according to your environment. For HPE employees, I would recommend using the native **Windows Secure Channel library**:



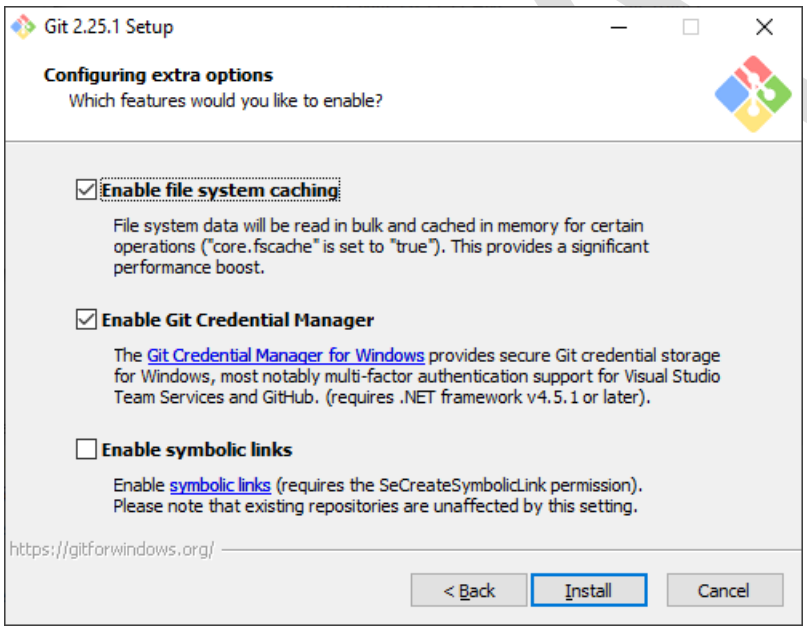
- Use the **Checkout Windows-style** as it is the recommended setting on Windows:



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



- Keep the default extra options



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

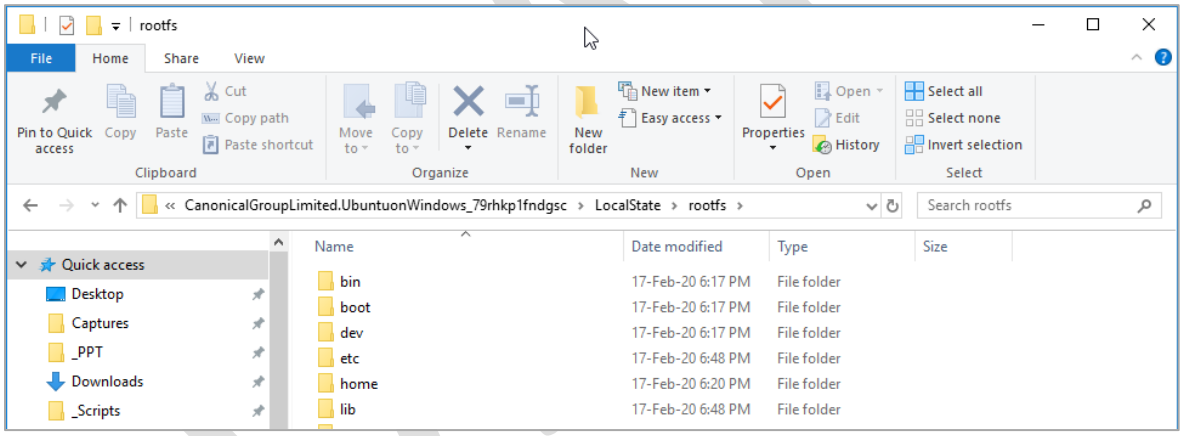
Where do my files live in my Ubuntu WSL?

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

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

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

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

> **Note**: Ubuntu root directory is located on your Windows file system in a hidden folder inside your User AppData directory:
> *`%USERPROFILE%\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_xxx\LocalState\rootfs\`*
>
> 
>

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

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

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

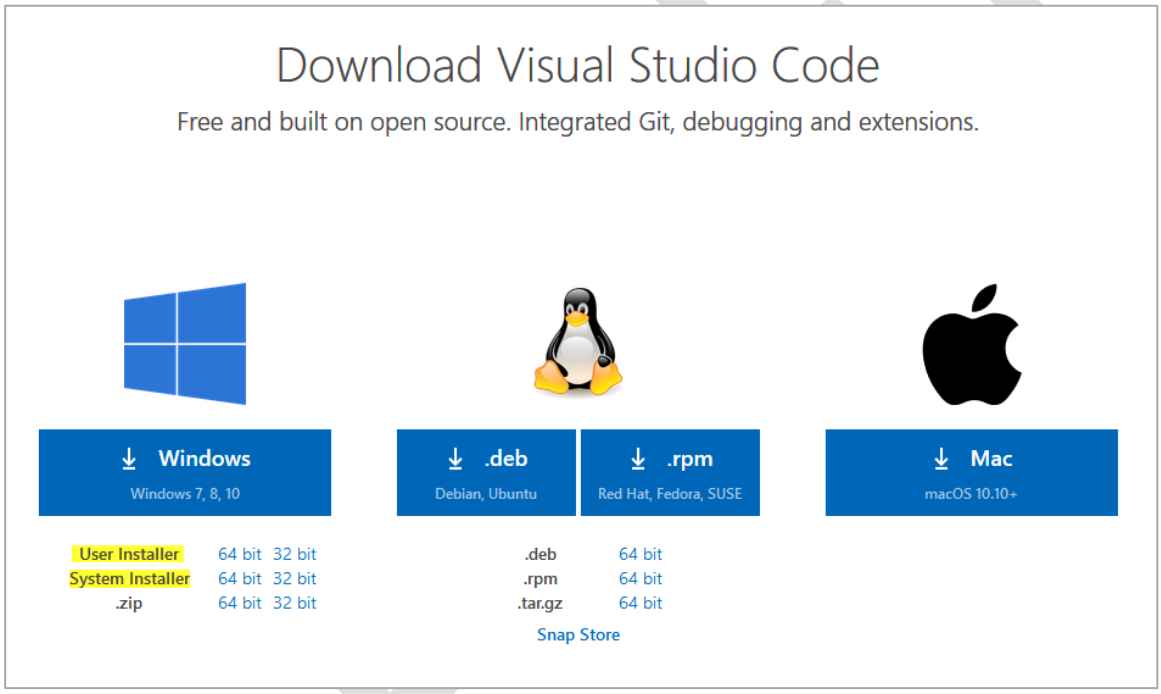
Chapter-3 – Preparing Microsoft Visual Studio Code

Installing Visual Studio Code

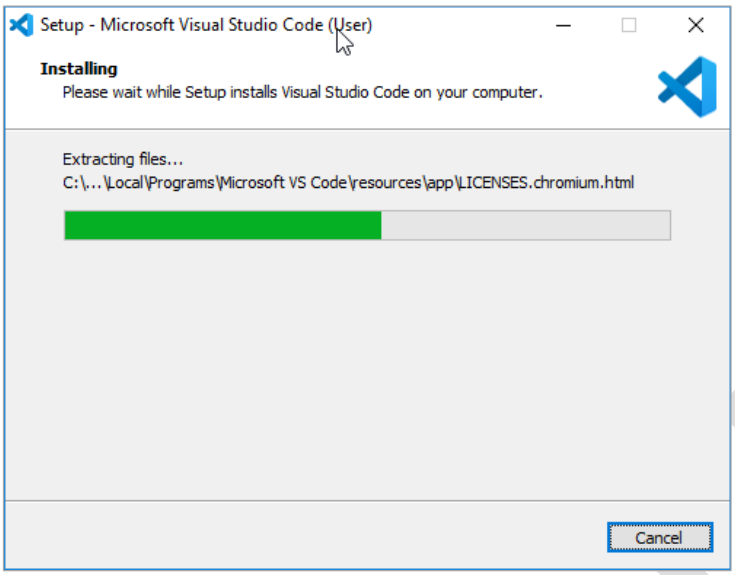
Visual Studio Code (also known as VS Code) is one of the most popular software for source-code editing developed by Microsoft. It includes many good features for debugging, it supports embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

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

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

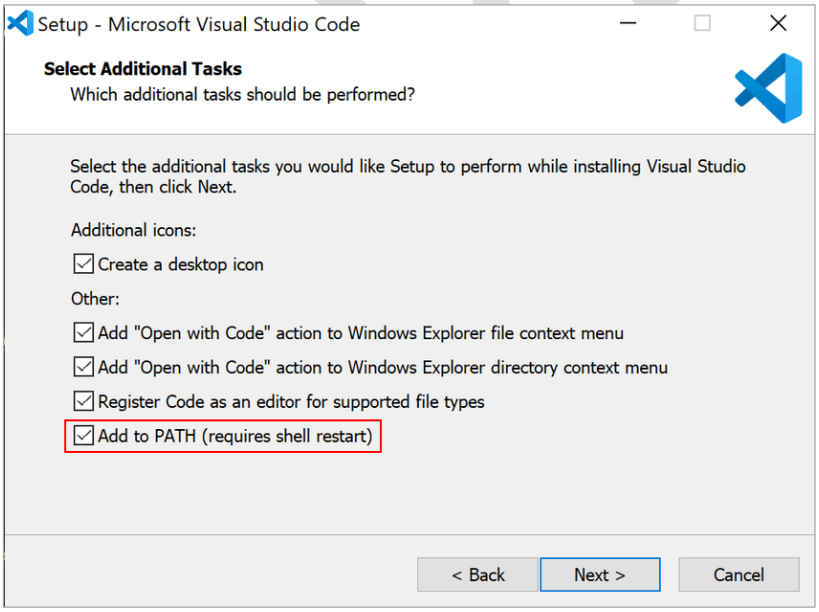


- You can proceed with the installation using all default parameters.



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

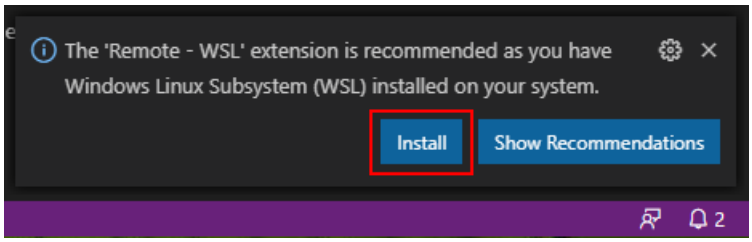
- When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily interact with WSL.



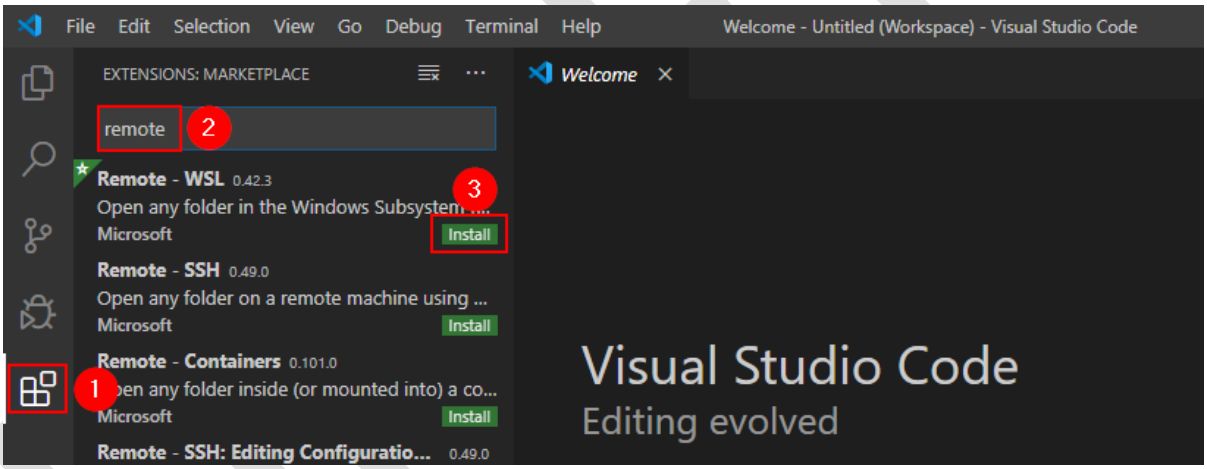
- When completed, launch **VS Code**.

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

- Click on **Install**

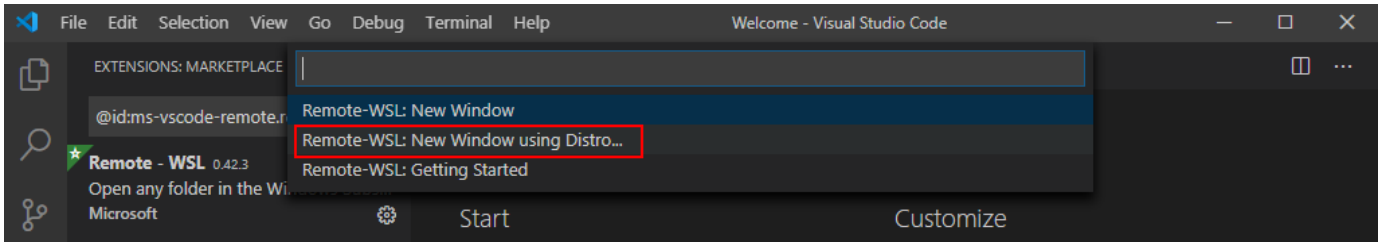


This is the extension that VS Code need to connect to our Ubuntu runnning in the Windows Subsystem for Linux.

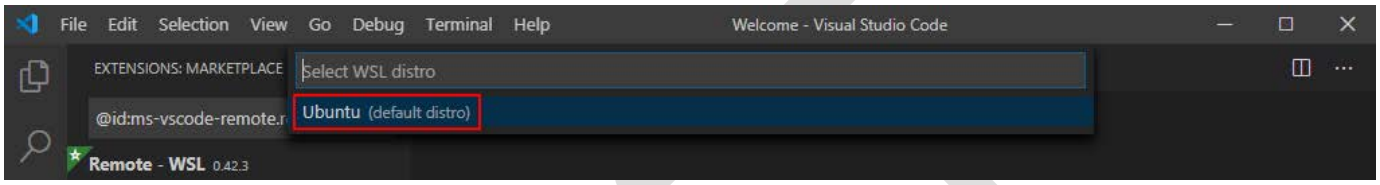
> **Note**: If the message is not appearing, you can install manually this extension by clicking on the **Extensions** icon on the Activity bar and search for **Remote – WSL**
>
> 
>

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

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

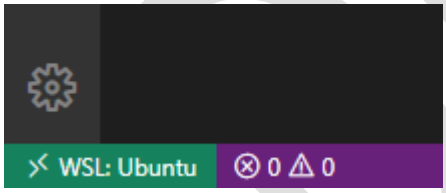


- Then select **Ubuntu**

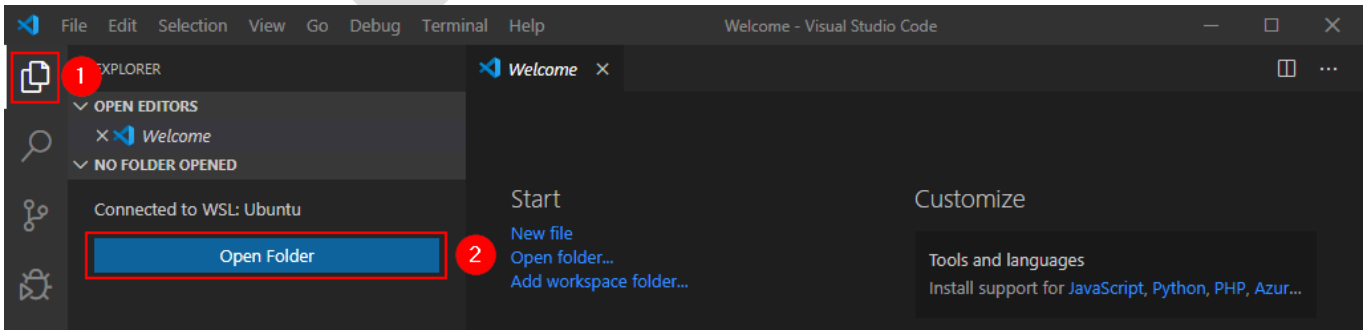


- A Remote connection to Ubuntu is taking place in a new VS Code window:

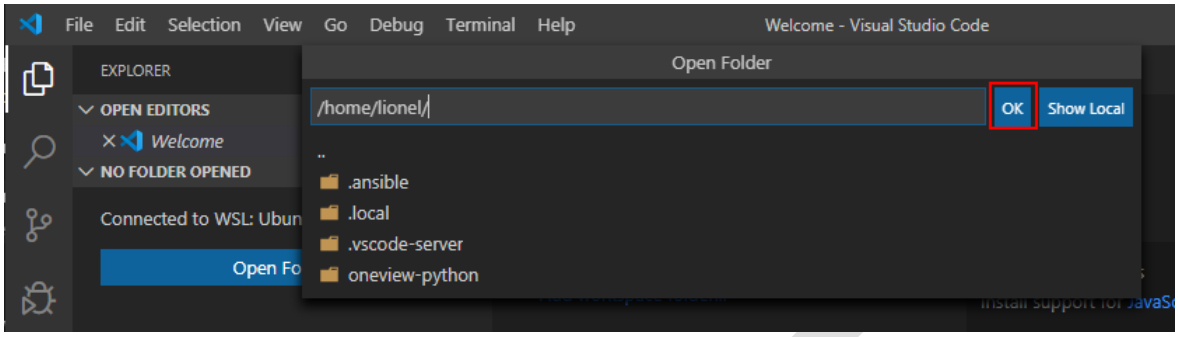
- You can close the previous VS Code window.
- Once connected, in the lower left corner of the Status Bar, you shoud see that you're connected to your WSL: Ubuntu instance.



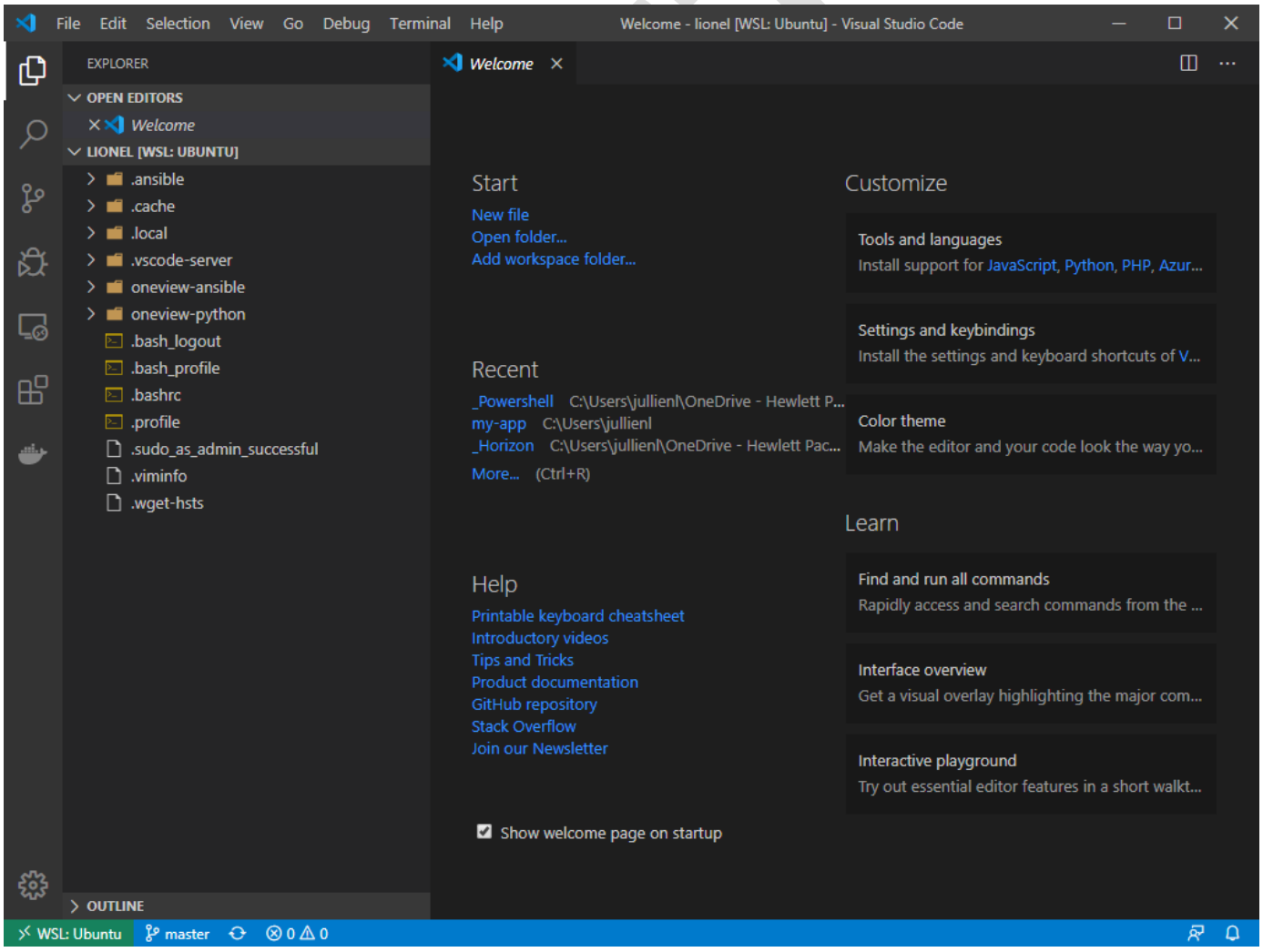
- You can then open a folder located on the WSL:Ubuntu using the **Explorer** icon on the Activity bar



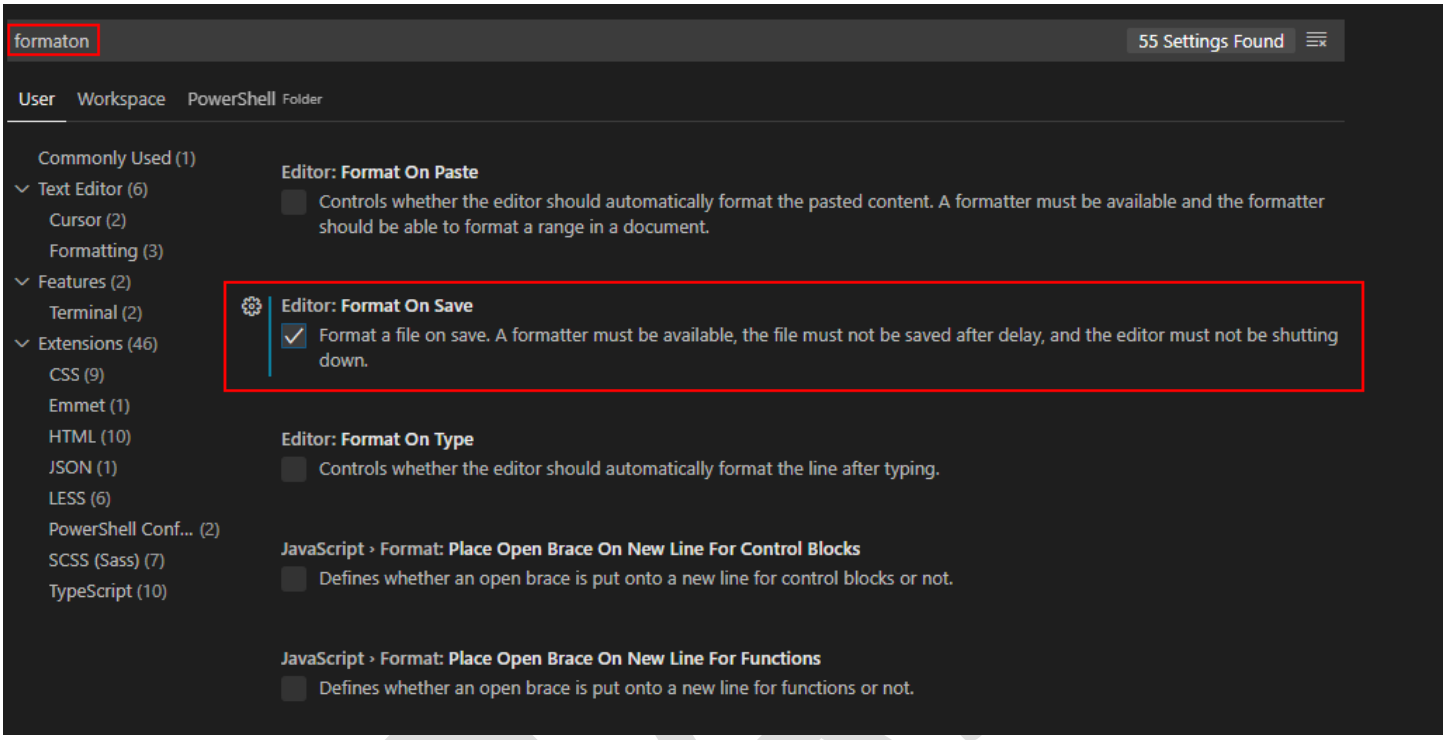
- Click **OK** to the `/home/<username>` folder



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

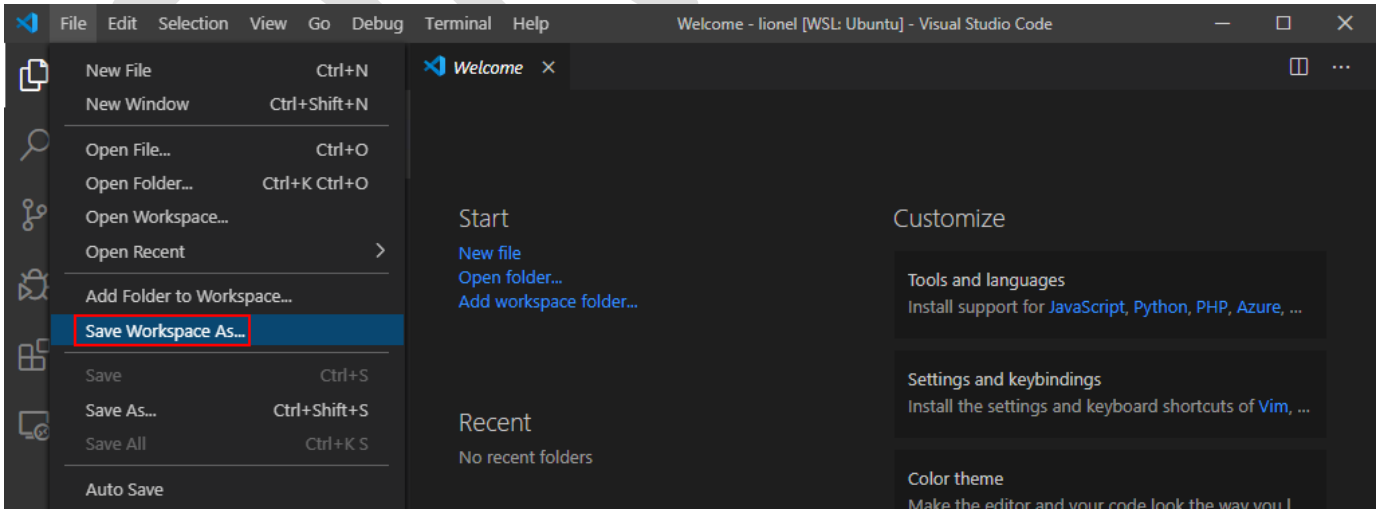


There are tons of VS Code settings that you can tweak but there is one that is really recommended when writing scripts is the **Format On Save**. This “must have” option automatically formats your code when you save your work by improving spacings, indents, by wraping lines if necessary and fixing quotes.

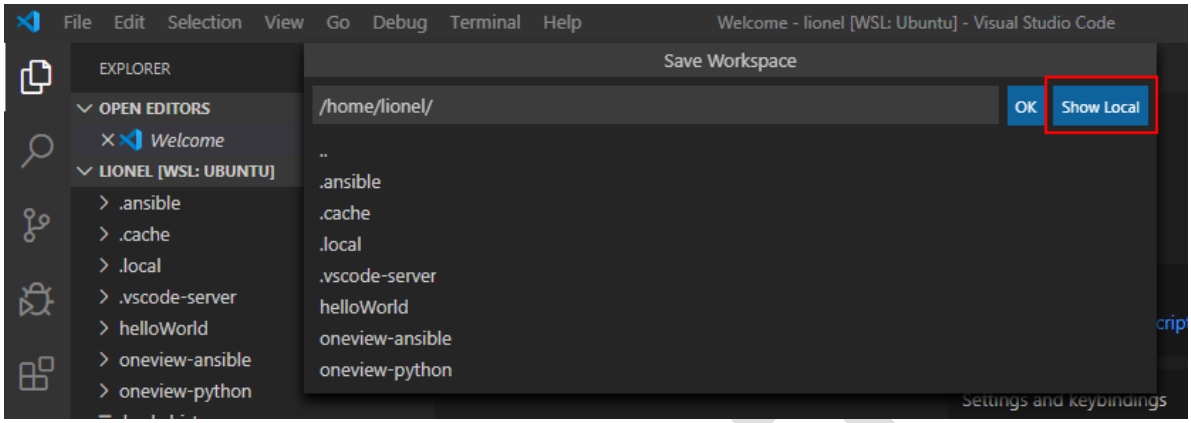


The next step is to save the current VS Code configuration in a workspace. A VS Code "workspace" is usually just your project root folder.

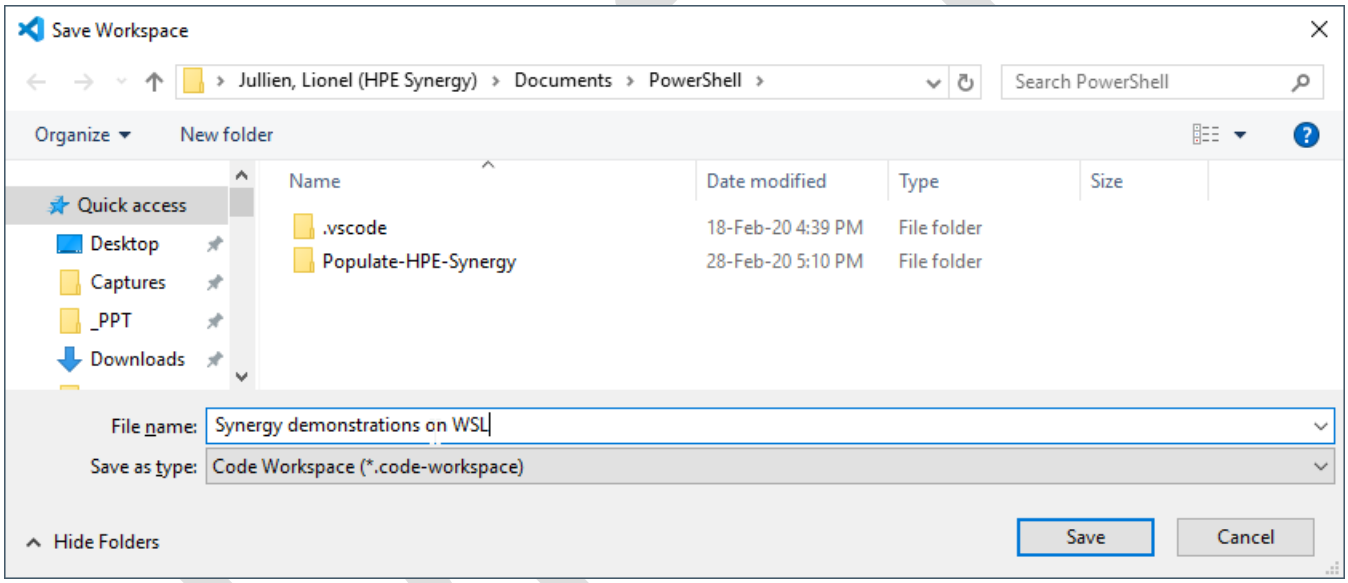
- Select **File** menu and click on **Save Workspace as**:



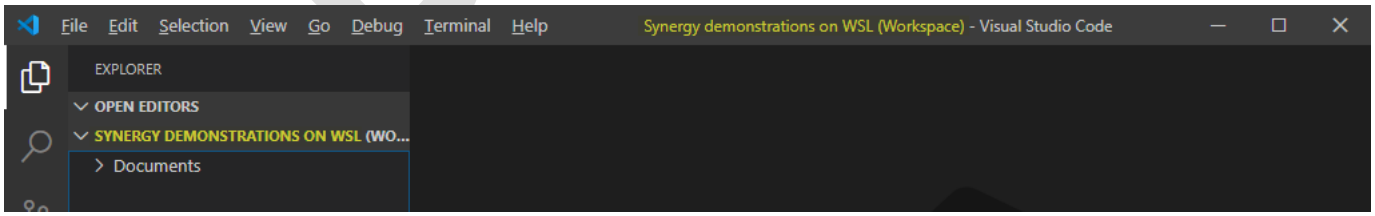
- Select **Show Local** and select a folder in your Windows Documents folder



- Select an emplacement and enter a name like **Synergy demonstrations on WSL** then click **Save**



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

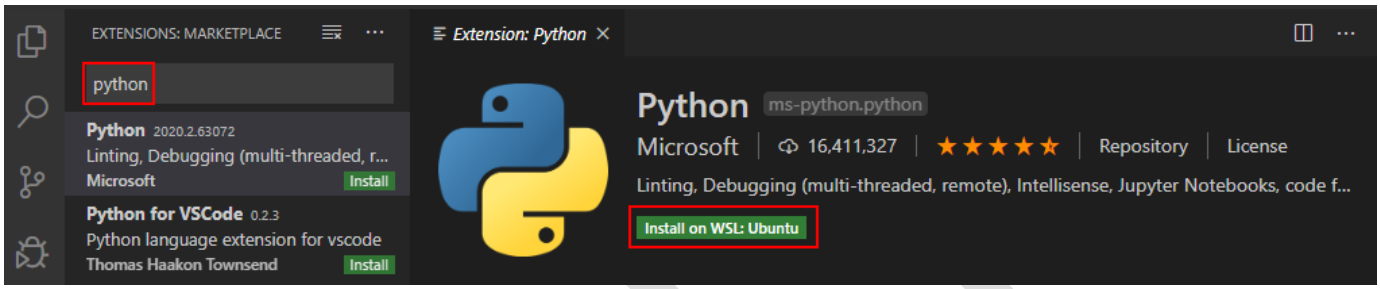


Preparing VS Code for Python

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

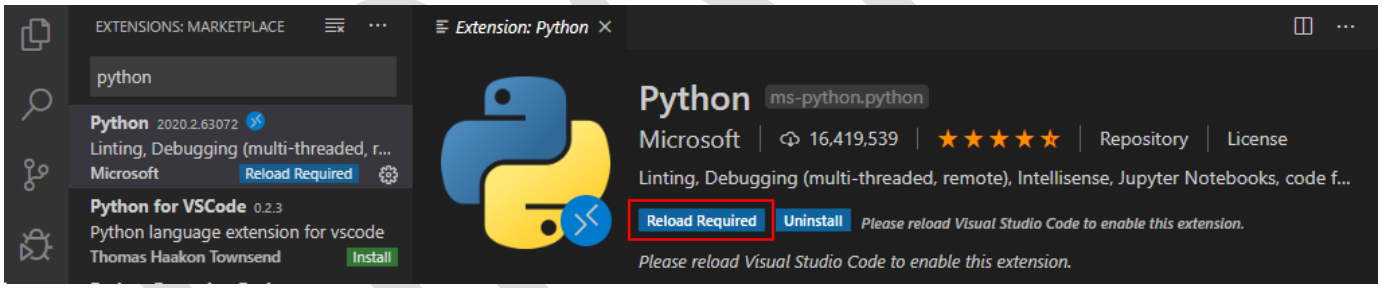
There are popular ones for PowerShell, Python, Ansible…

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



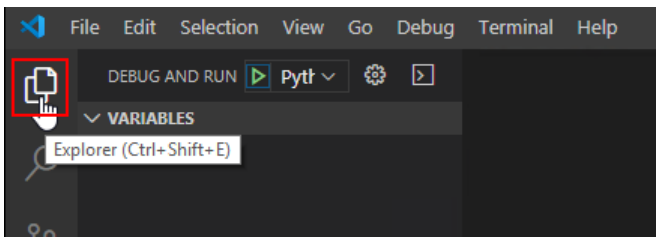
This extension provides nice advanced features like Syntax highlighting, IntelliSense, debugging, code formatting, access to module documentation, ability to run the active script file in the VS Code terminal console, etc.

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

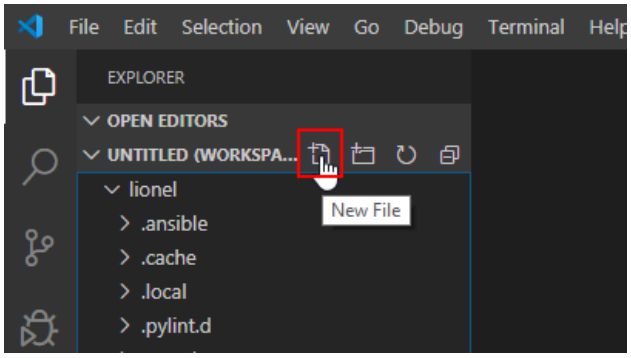


> **Note**: For the Python extension, we are going to use the Python interpreter running in our Ubuntu Linux distribution hosted by WSL. Another good option if you don't want to use WSL would be to run Python directly on your Windows system, see https://www.python.org/downloads/windows/ and https://docs.microsoft.com/en-us/windows/python/beginners

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

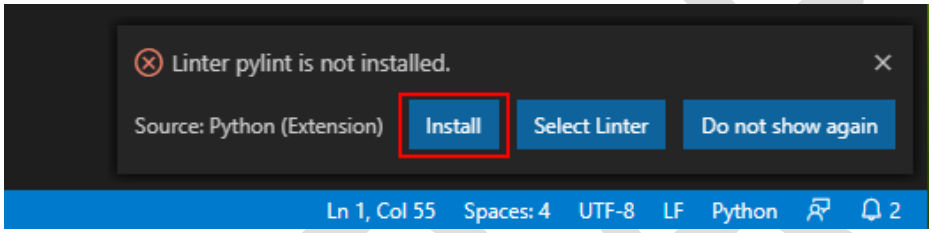


- Then click on **New File**



- Enter a name like **test.py** then press **ENTER**

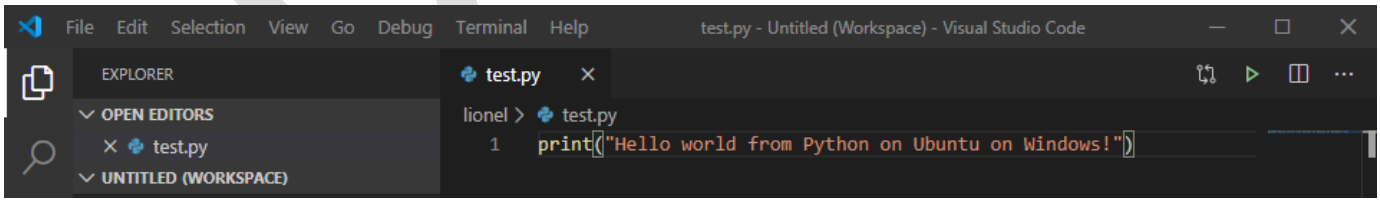
- You can install pylint by clicking on **Install** when the window pops-up at the creation of the Python script. A code linter is a program that analyses your source code for potential errors and code style guideline violations. Pylint is one of the well-known Python linter programs similar to Pychecker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



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

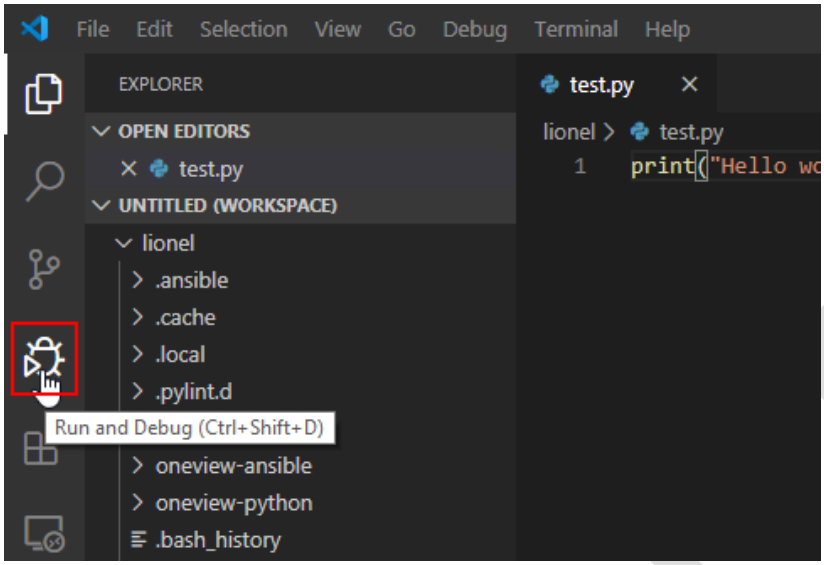
- Enter the following line to build the Python script test.

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

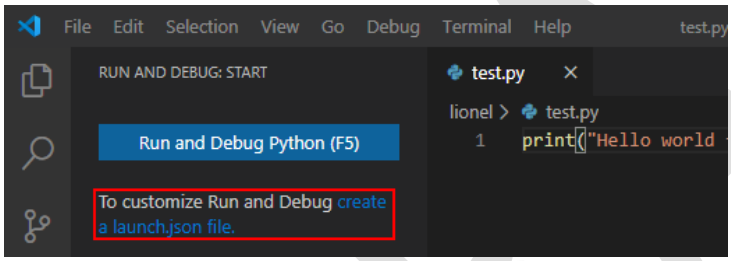


- Save the file by pressing **CTRL** + **S**

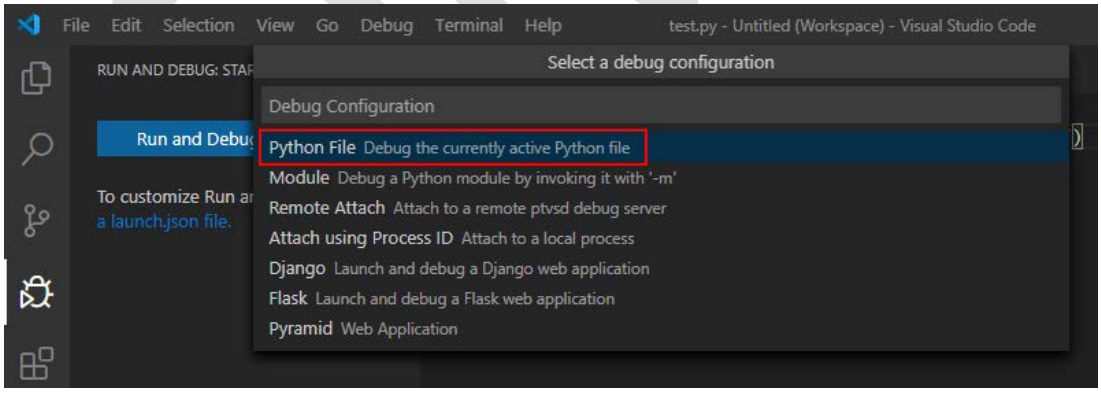
- Click on **Run and Debug**



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

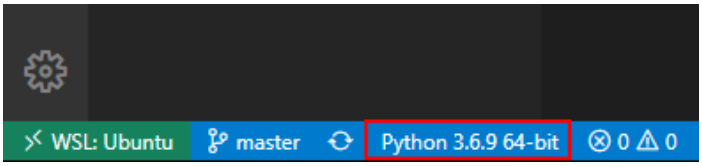


- Select **Python File**

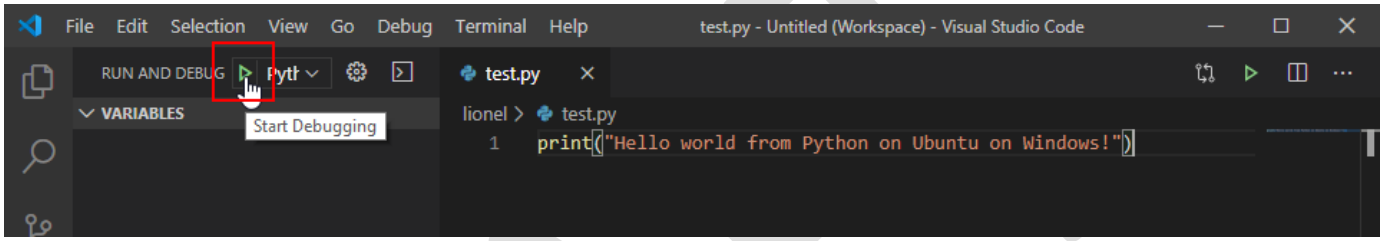


- To complete the Python debug and Run configuration, press **CTRL** + **S** to save the configuration file then **CTRL** + **F4** to close the configuration file.

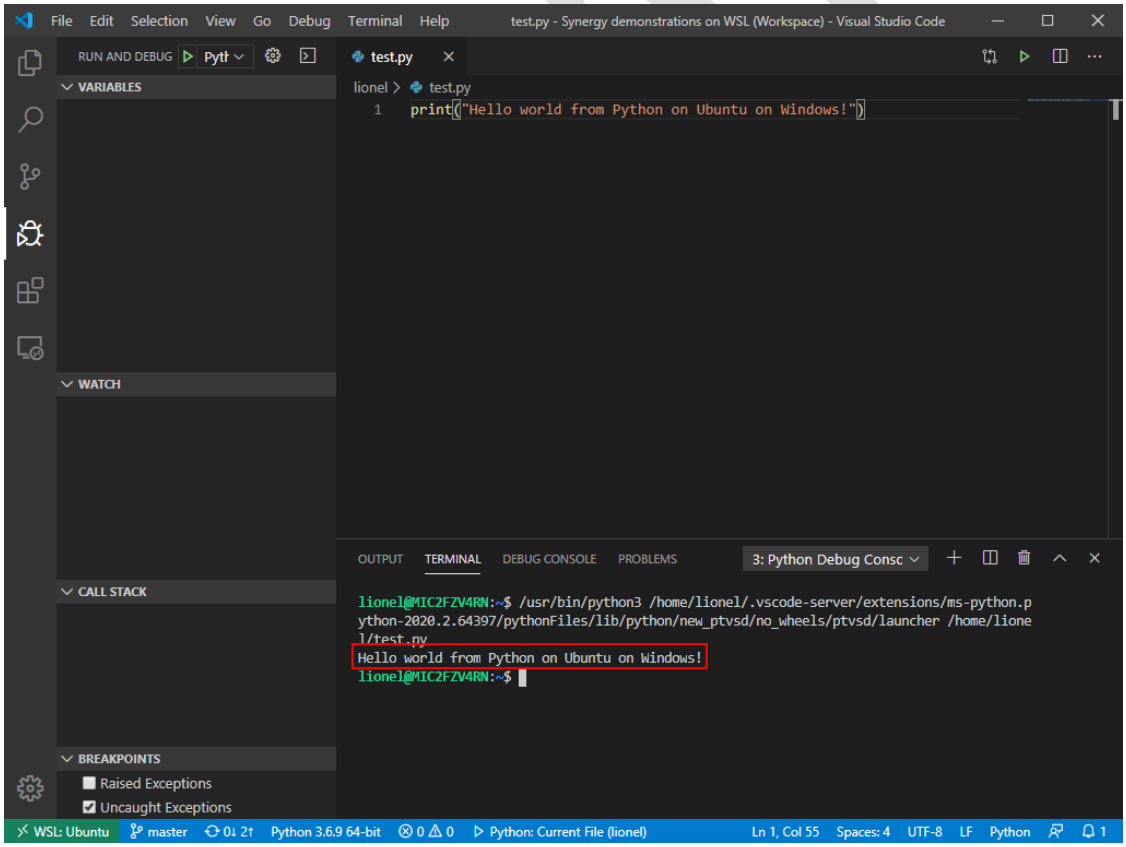
> **Tip**: You can see the Python interpreter version in use in the status bar



- Now click on **Start Debugging**



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

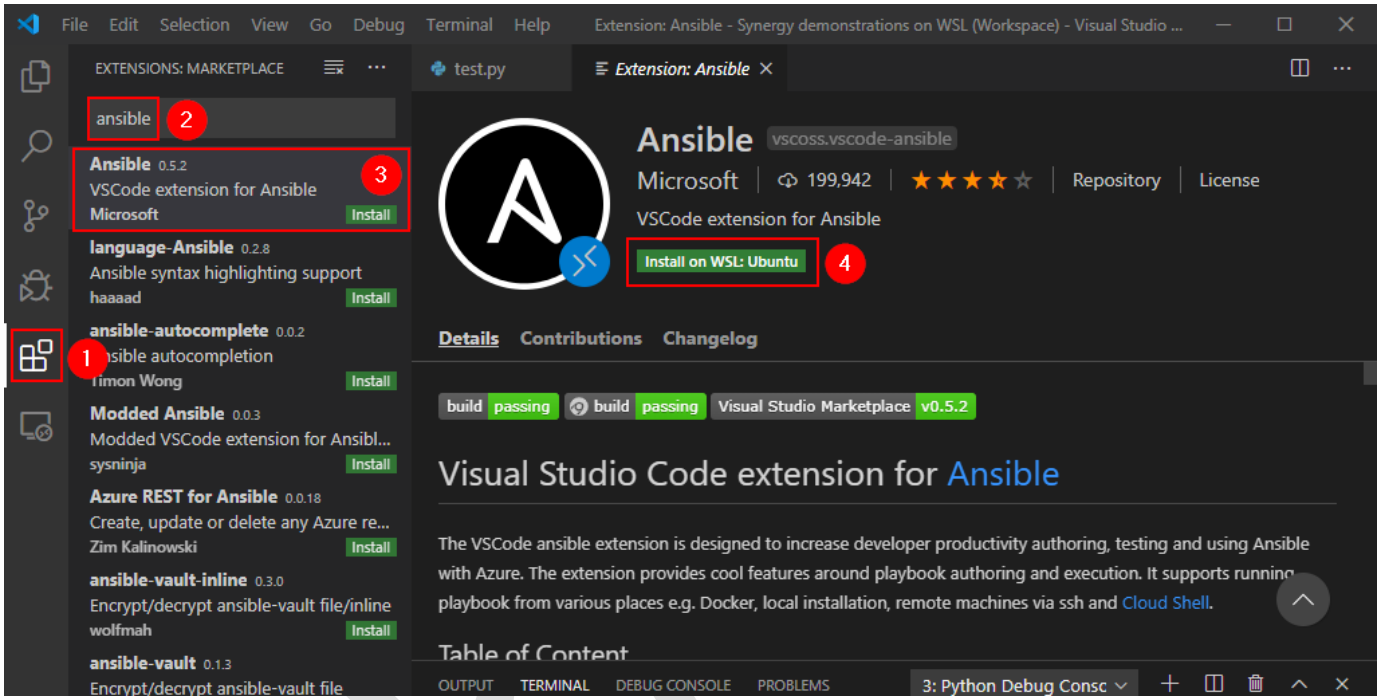


We are now all set for Python scripting and debugging.

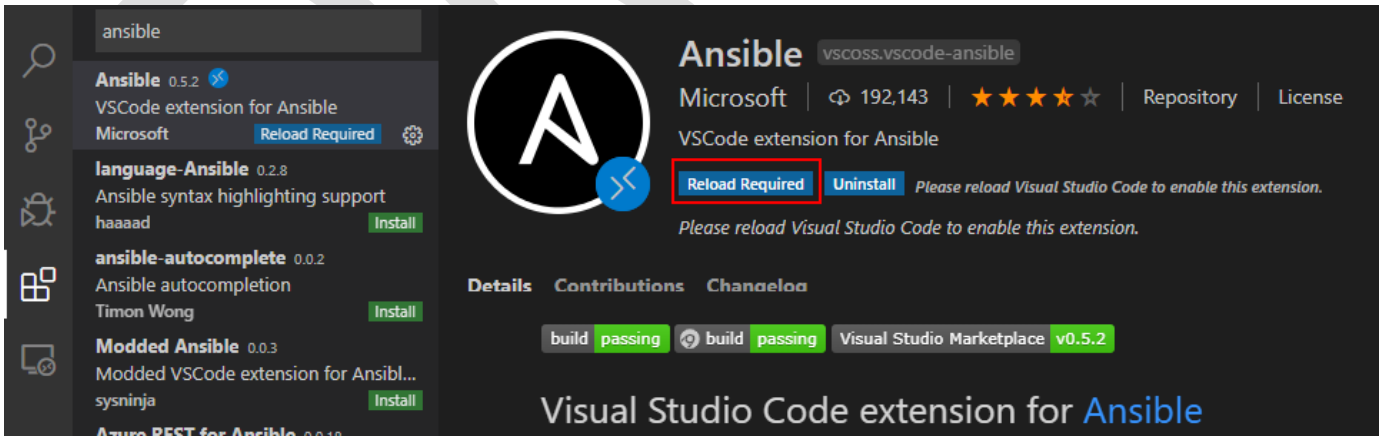
Preparing VS Code for Ansible

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

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



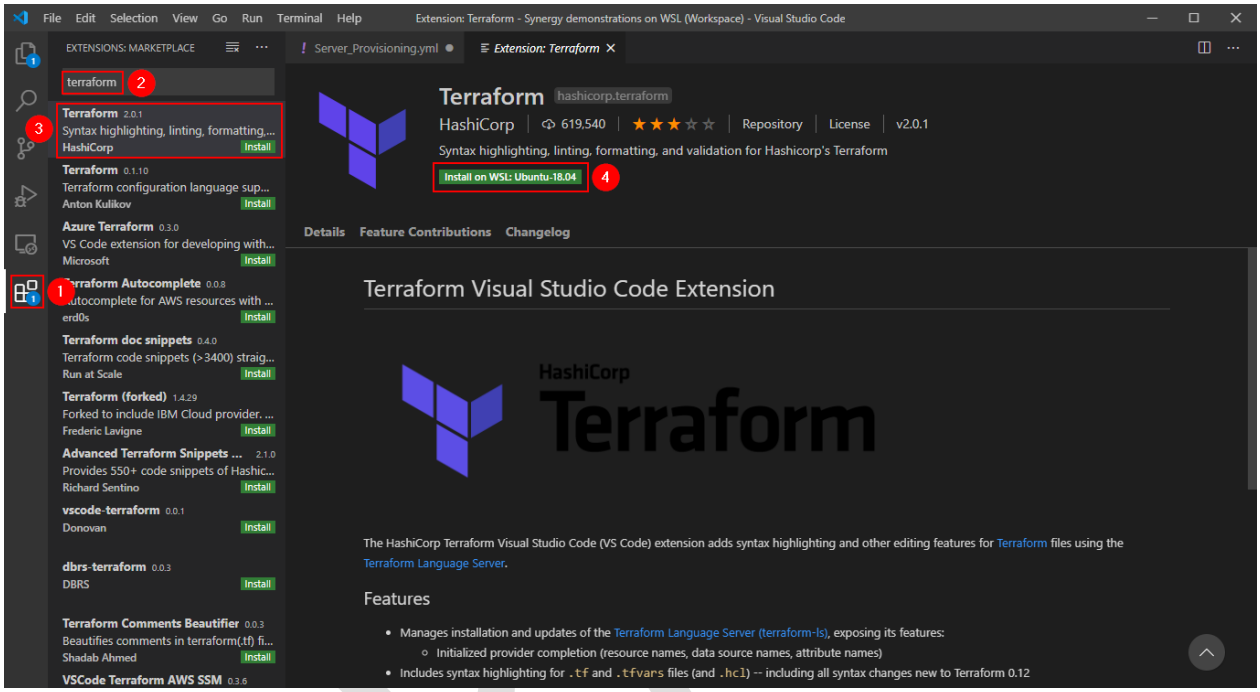
- Once the installation is complete, click on **Reload Required** to activate the extension.



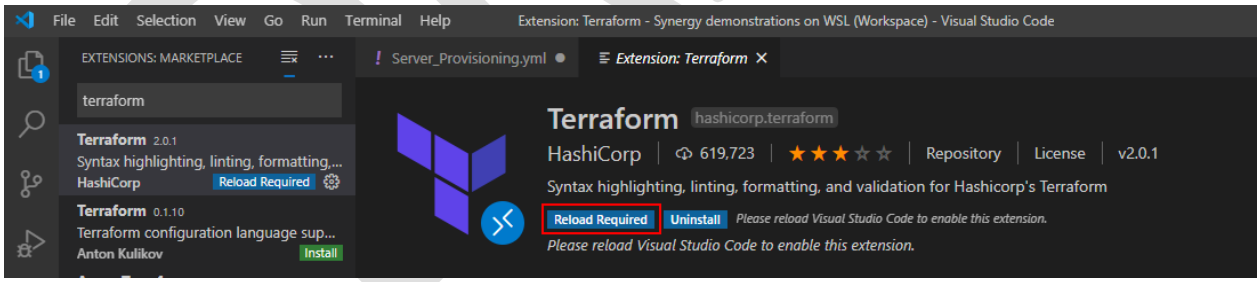
Preparing VS Code for Terraform

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

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



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



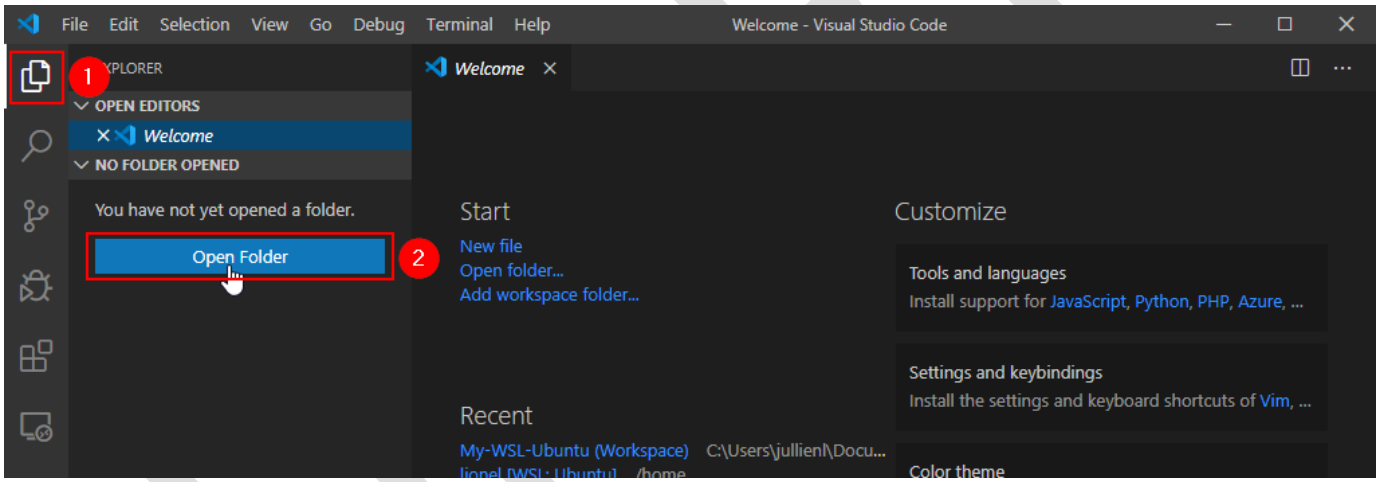
Preparing VS Code for PowerShell

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

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

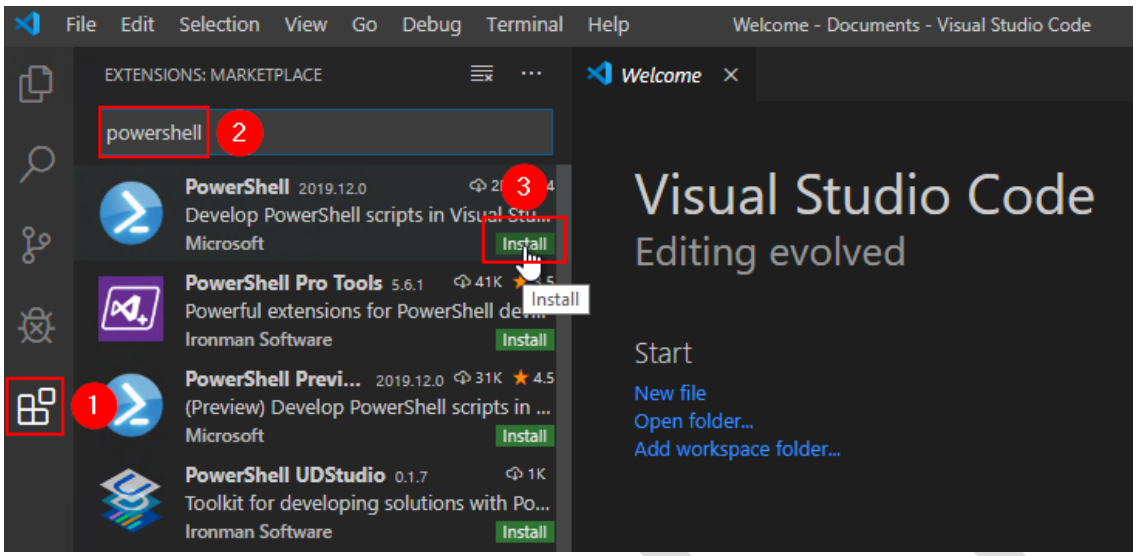
- Open a new VS Code instance by clicking on the Visual Studio Code icon on your desktop:

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

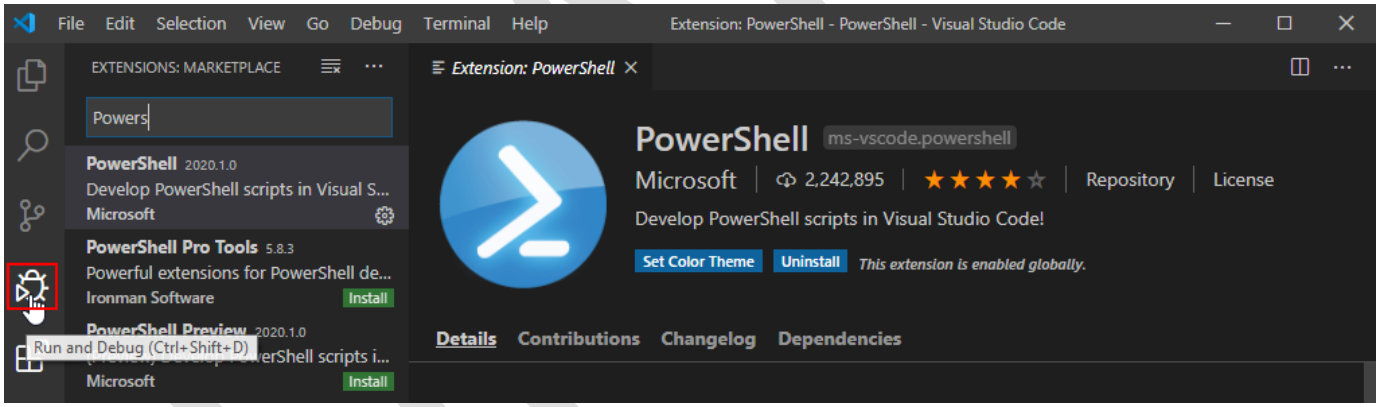


- Open the folder you want your PowerShell project to be in (e.g. \Documents\PowerShell) and click **Select Folder**.

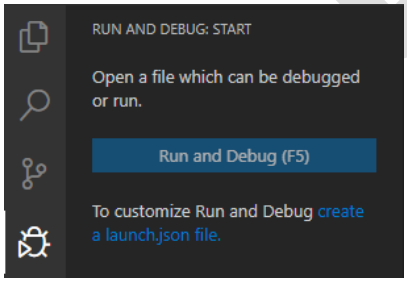
- Go to the **Extensions** pane and search for the **PowerShell** extension and click on the **Install** button:



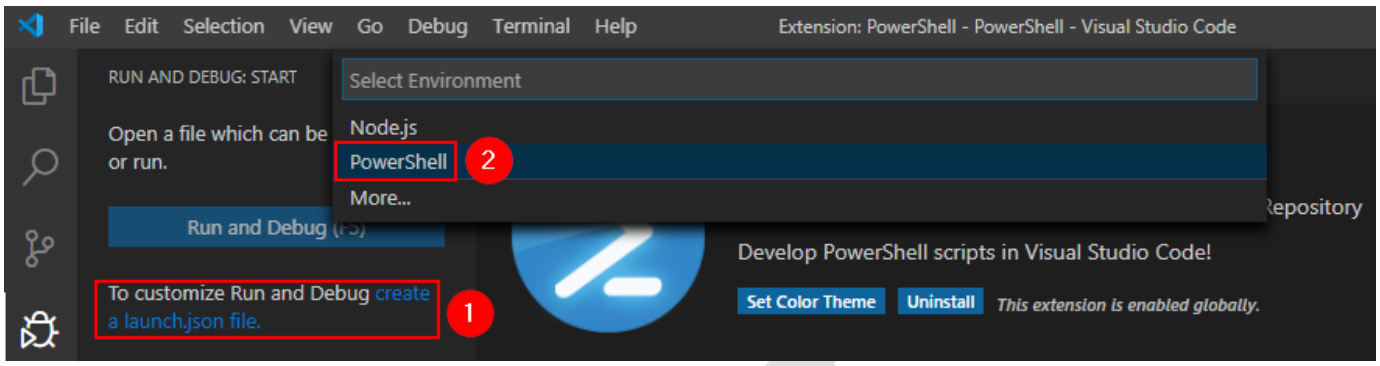
- To configure the PowerShell interpreter once the extension is installed, click on **Run and Debug** on the Activity Bar



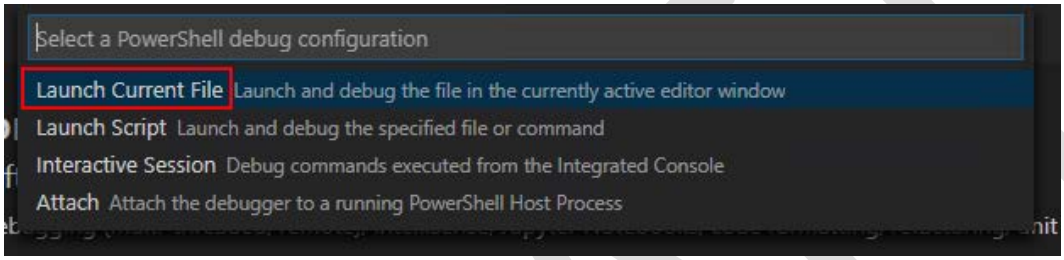
Visual Studio Code's built-in debugger needs some quick setup to run and debug PowerShell scripts. If your system is already configured, you can skip this section. When VS Code is not configured for PowerShell, the following dialog is displayed:



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

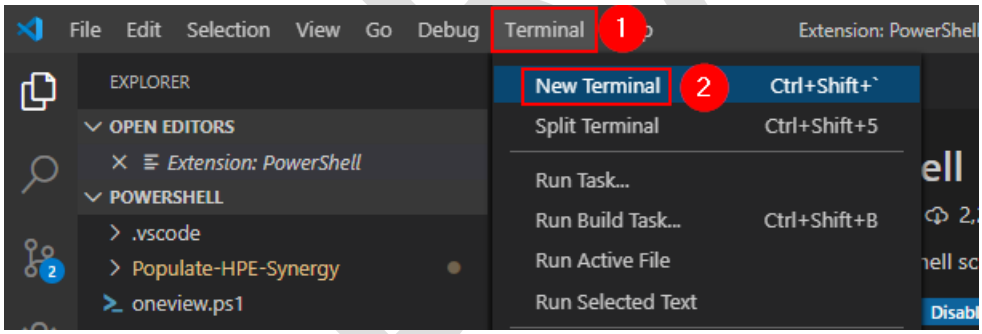


- Then select **Launch Current File**

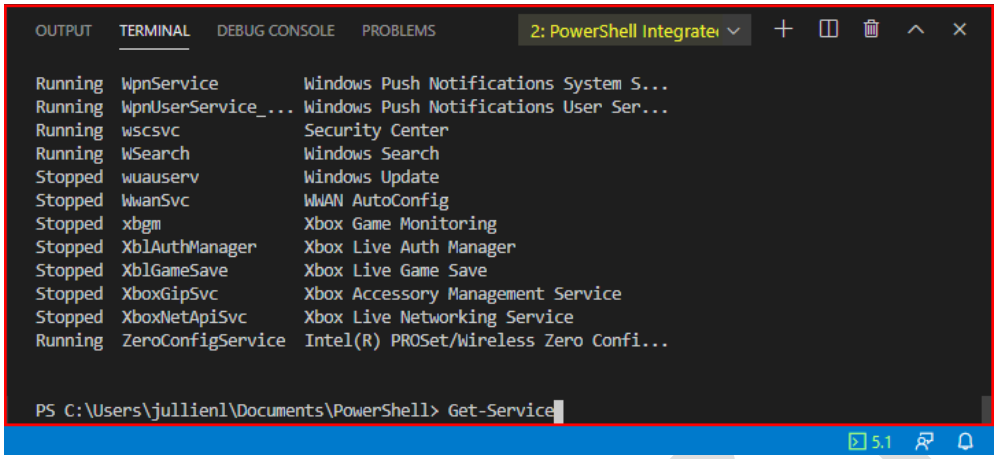


- Then press **CTRL** + **S** to save the configuration file then **CTRL** + **F4** to close the configuration file.

- Select now **Explorer** in the activity bar, a PowerShell Integrated console should start, or you can select **Terminal** / **New Terminal**

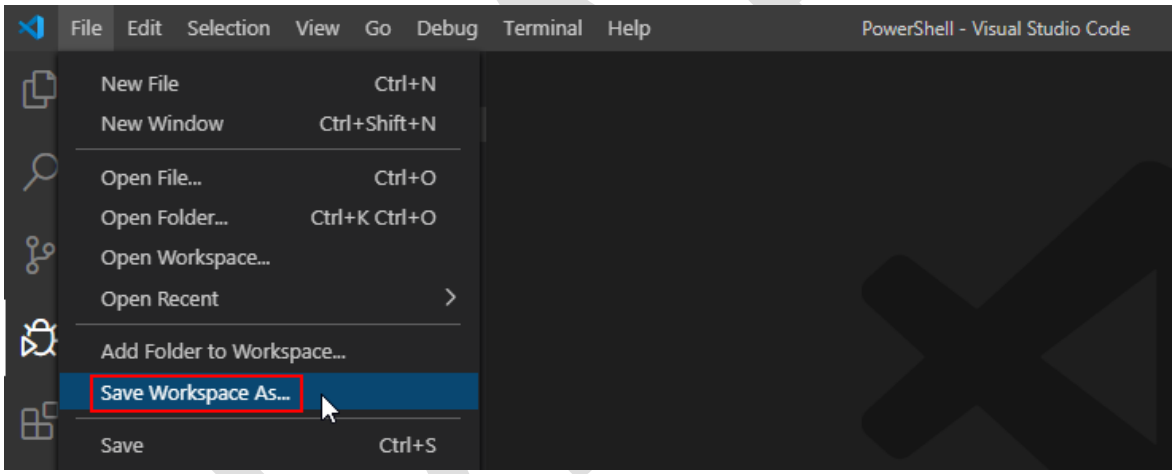


- You can enter **get-service** to make a quick test:

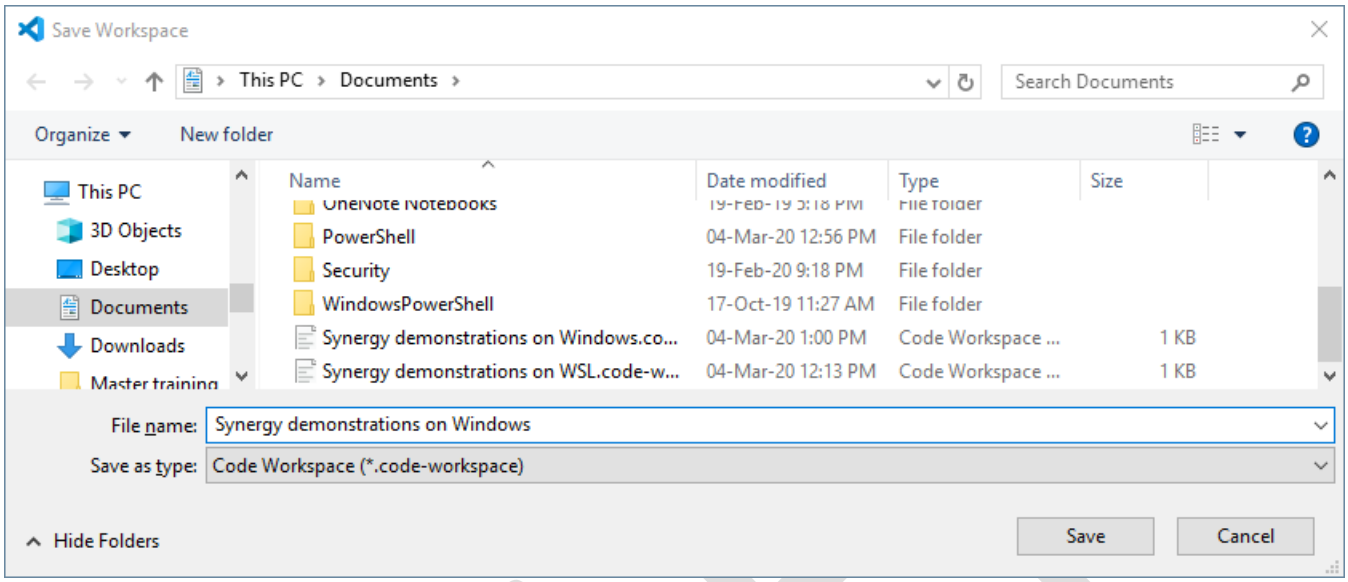


Last step is to save the workspace as our PowerShell workspace project.

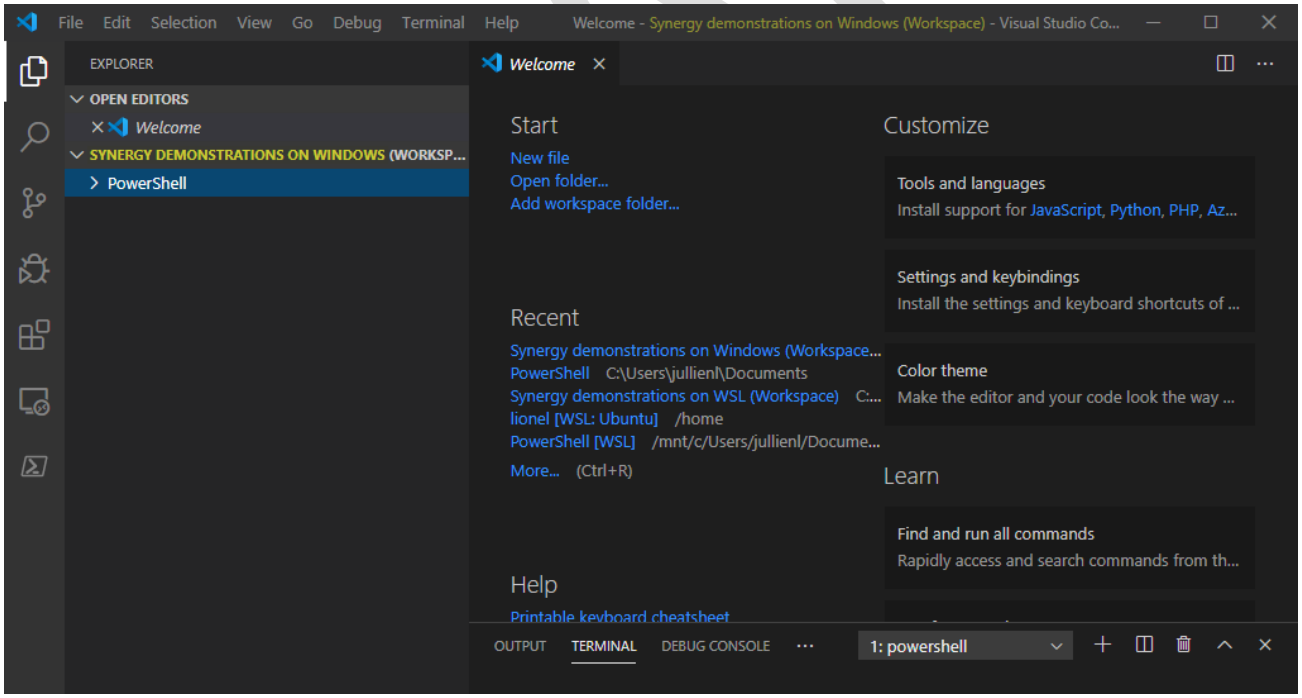
- Select the **File** menu then select **Save Workspace As...**



- Select your Windows User Home directory, enter a project name like **Synergy demonstrations on Windows** then click **Save**



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

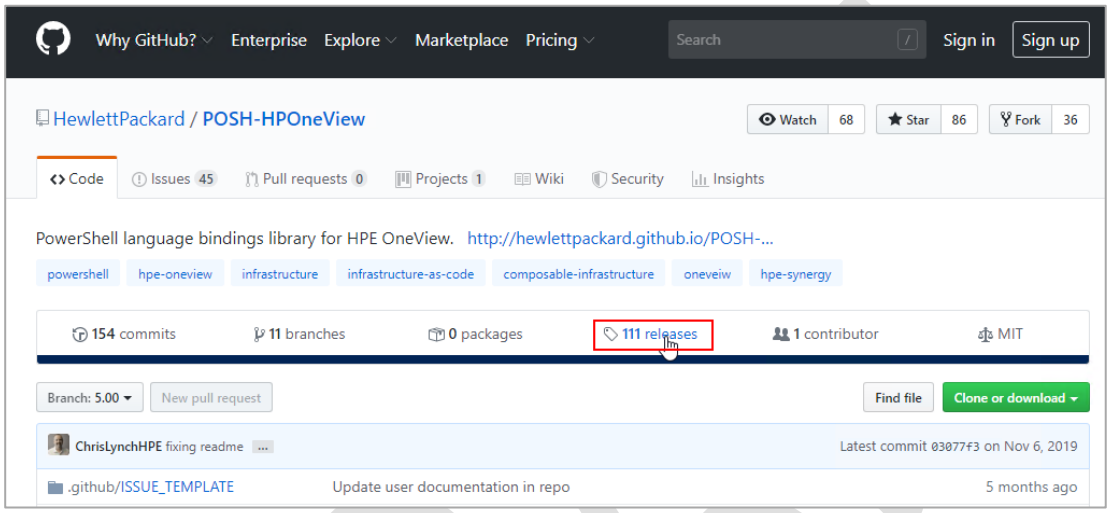


- You can also set the **Format on Save** option in **Preferences** to automatically format your code on save.

Installing the PowerShell library for HPE OneView

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

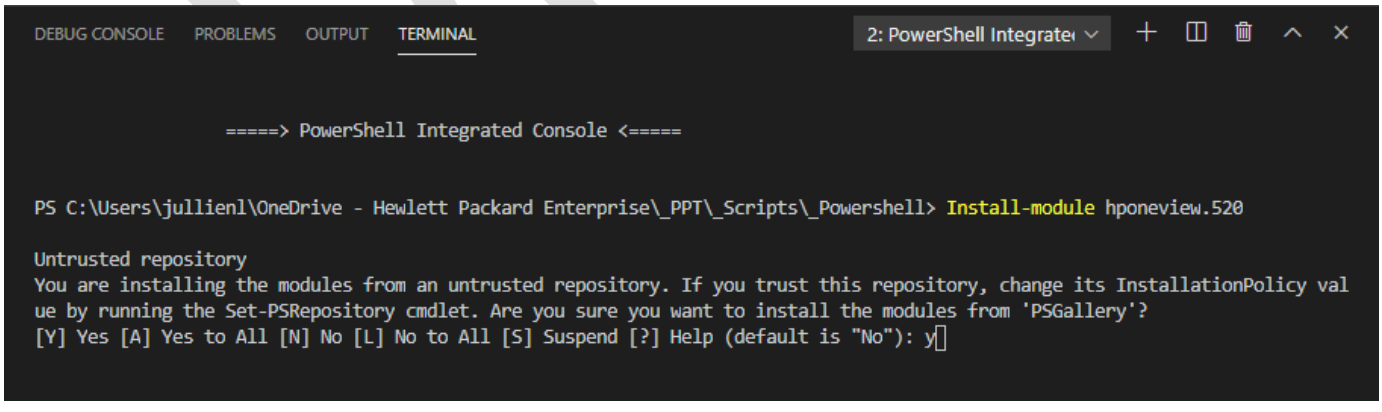
- Open a browser and visit https://github.com/HewlettPackard/POSH-HPOneView
- Select **Releases**



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

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

```
# Install library from the PowerShell Gallery
Install-Module HPOneView.520
```



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

- You can then Import the module using:

```
Import-Module HPOneView.520
```

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

```
get-hpovversion
```

```
PS C:\Users\jullienl\OneDrive - Hewlett Packard Enterprise\_PPT\_Scripts\_Powershell> Get-HPOVVersion


LibraryVersion Path
-------------- ----
5.20.2422.3962 C:\Users\jullienl\OneDrive - Hewlett Packard Enterprise\Documents\WindowsPowerShell\Modules\hponeview.520...
```

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

> **Note**:
> If you get the following error:
> `Import-Module: The library is unable to load due to this system missing the required .Net Framework 4.7.2 client.`
> You need to install .Net Framework 4.7.2 or later! You can download .Net Framework 4.8 from https://dotnet.microsoft.com/download/dotnet-framework/net48. Once installed, retry the import operation.
>
> **Note**:
> If you get the following error:
> `import-module: File <…>.psm1 cannot be loaded because running scripts is disabled on this system`
> You need to change your system policy to allow scripting. You can enter the following commands:
> `Set-ExecutionPolicy -ExecutionPolicy unrestricted`
> or
> `Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass -Force -Confirm:$False`
>
> **Note**: if you need to upgrade from a previous version of the PowerShell library for HPE OneView, refer to the Upgrading your HPE Synergy demonstration environment chapter.

This completes the PowerShell configuration and concludes Chapter-3

In the next chapter, we will configure HPE OneView.

Chapter-4 –Initial Configuration of the Demonstration Appliance

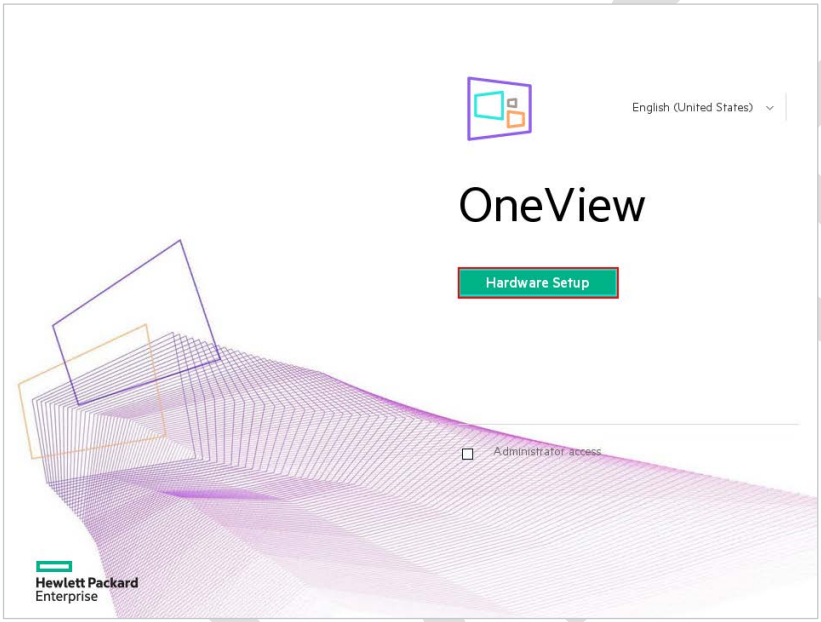
The installation of the scripting languages and the preparation of the Ansible environment is now complete, and OneView has certainly finished booting, we can now finalize the configuration of the appliance.

Hardware discovery and initial appliance configuration

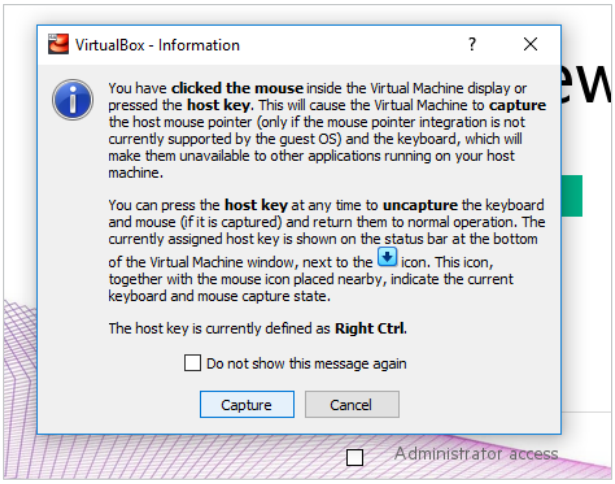
- Close all open applications running on your computer.

> **Important notice**: It is recommended that you close all programs before starting Hardware Setup as the lack of CPU/memory resources during this phase can produce negative effects on the appliance.

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



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

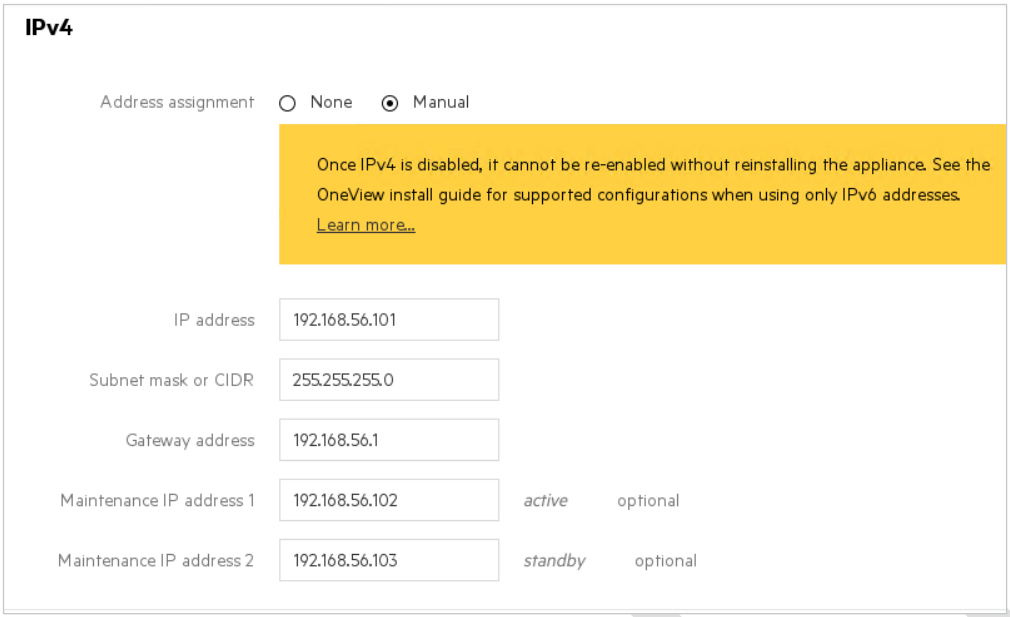


- Change the appliance name with **OneView.net**
- Then we can configure three static IP addresses taken from the "Host-only" VirtualBox subnet (192.168.56.0/24)

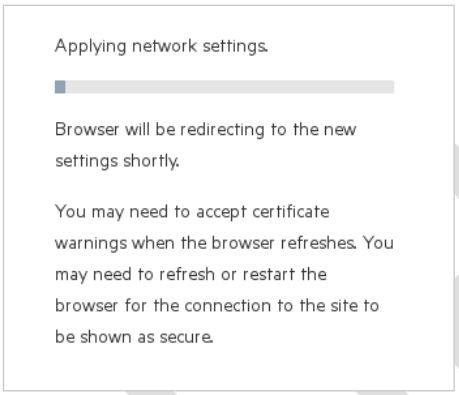
> **Note**: you can open a Windows command prompt and use `Ipconfig` to see the VirtualBox Host-only network adapter subnet

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

Windows IP Configuration


Ethernet adapter Ethernet0:

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

Ethernet adapter VirtualBox Host-Only Network:

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

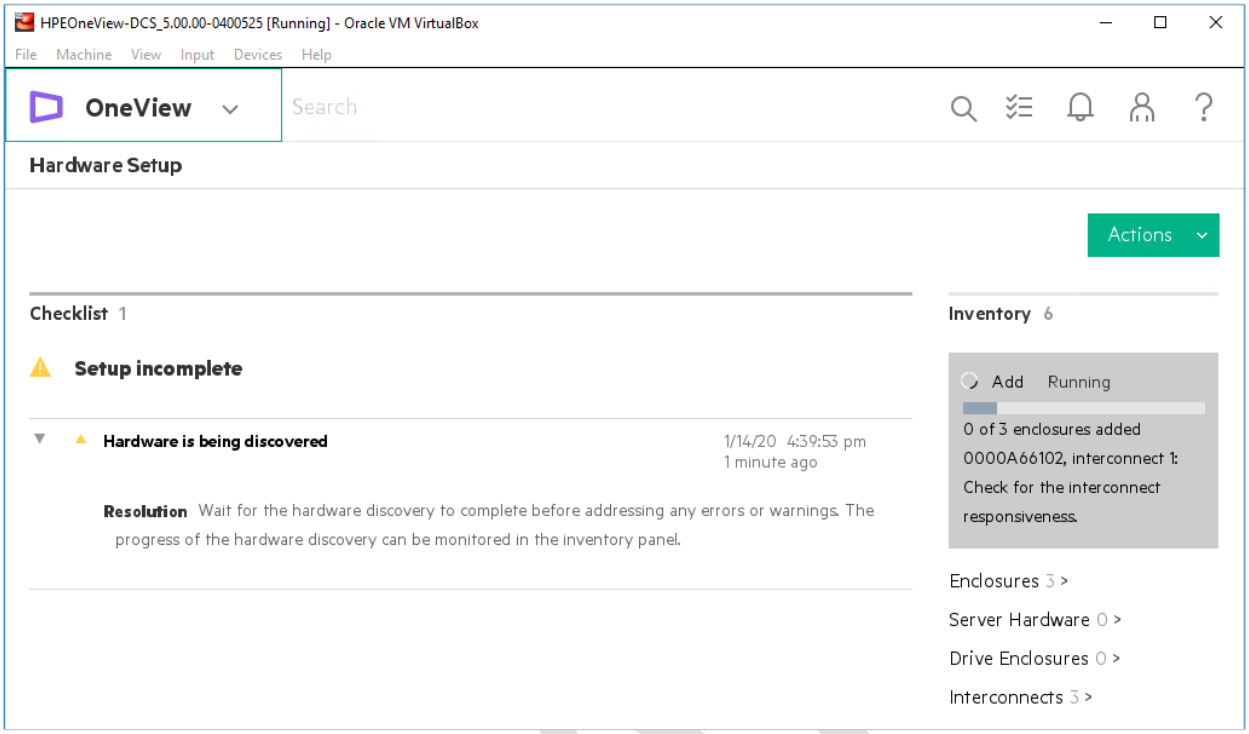
- We can enter the following IP addresses for the default "Host-only" VirtualBox subnet:
  - Primary IP address: **192.168.56.101**
  - Subnet mask: **255.255.255.0**
  - Gateway address: **192.168.56.1**
  - Maintenance IP1: **192.168.56.102**
  - Maintenance IP2: **192.168.56.103**



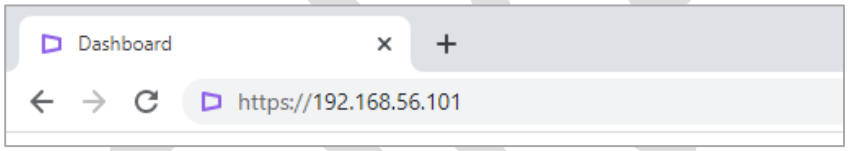
- Then click **OK**



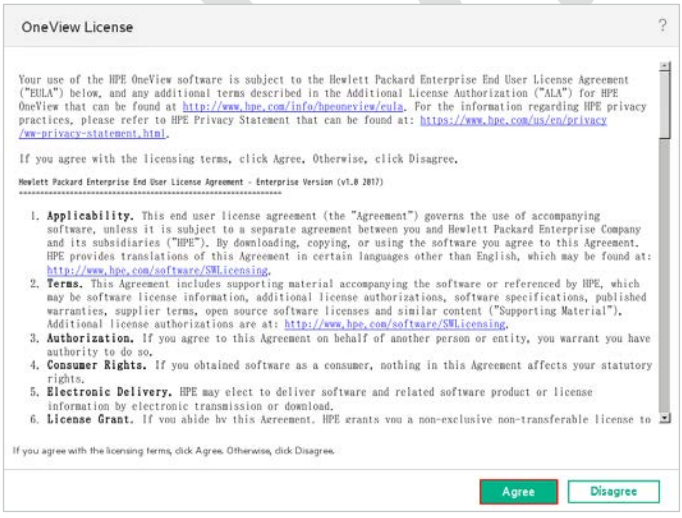
- Once network configuration completes, the initial round of hardware discovery starts:



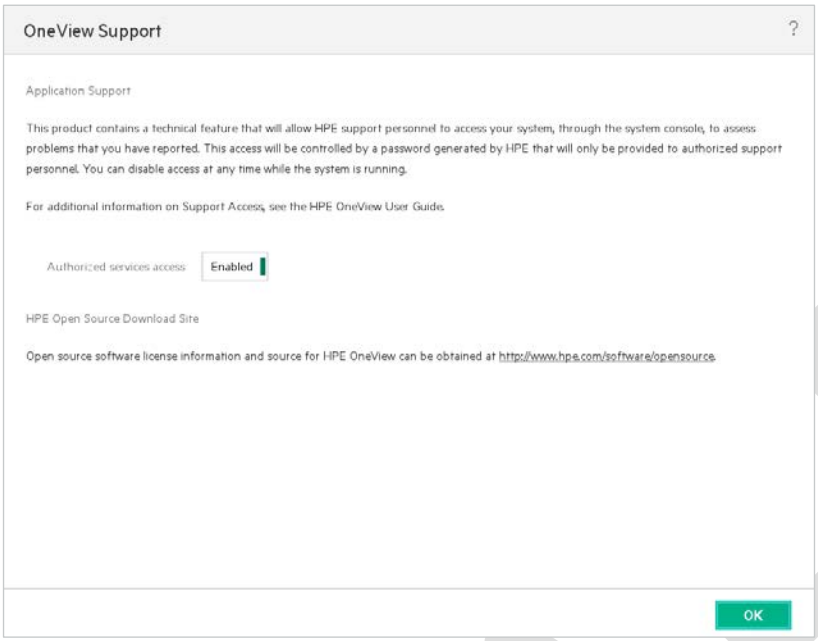
- At this point, you can access your demonstration appliance via a web browser using https://192.168.56.101



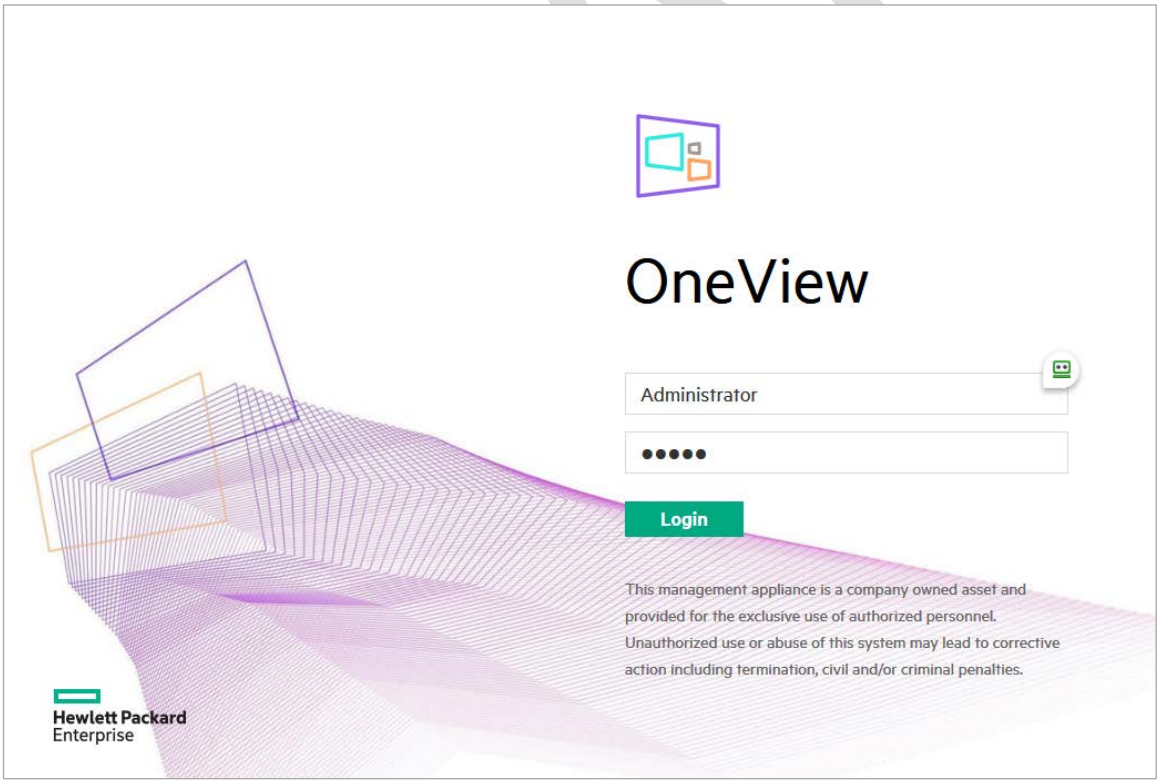
- Accept the OneView license agreement



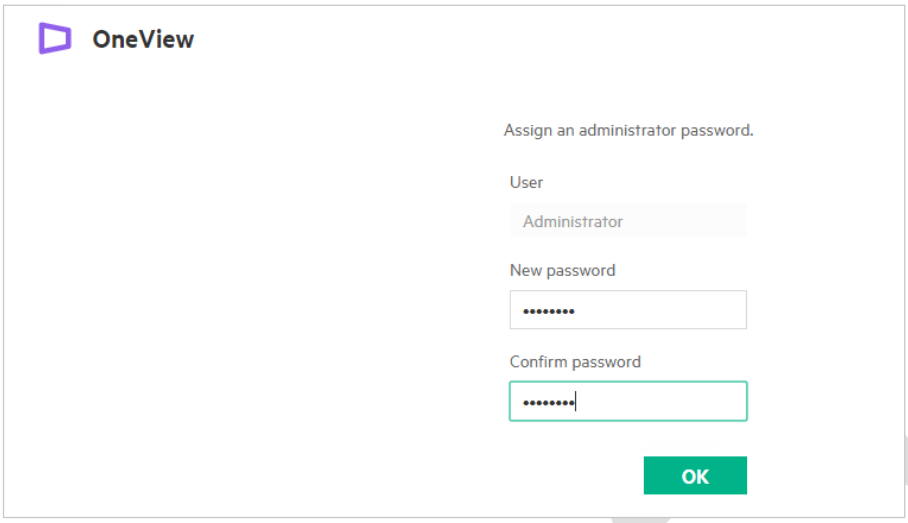
- Leave OneView Support enabled and select **OK**



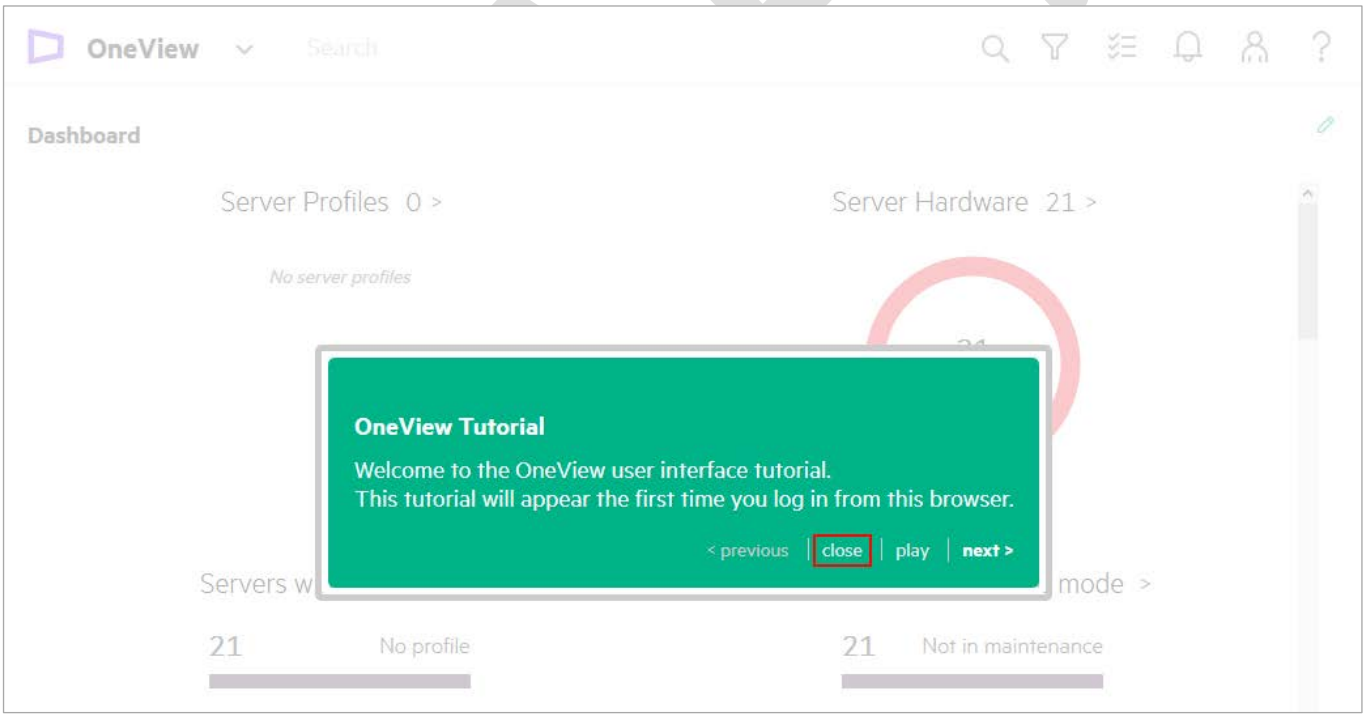
- Log in using **Administrator** / **admin**



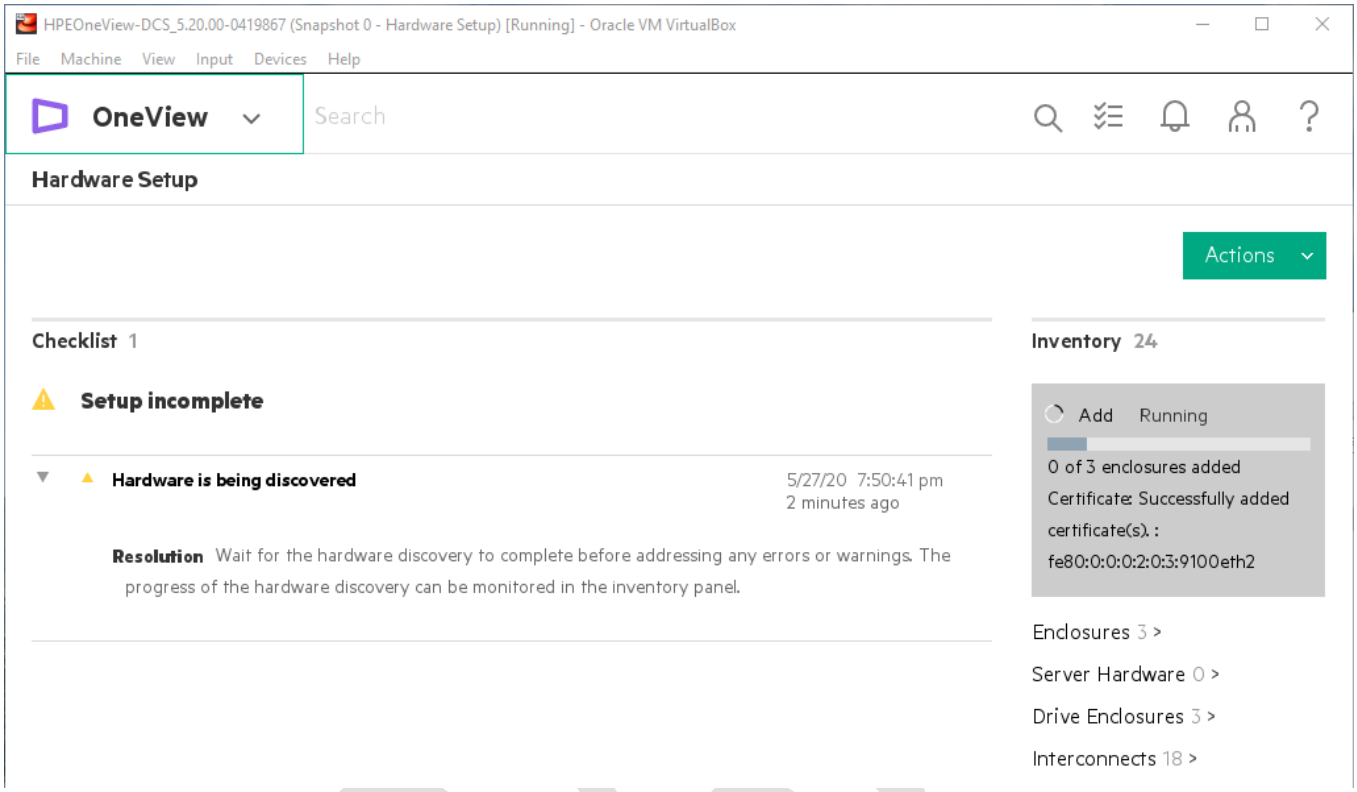
- Then change the default password to **password**



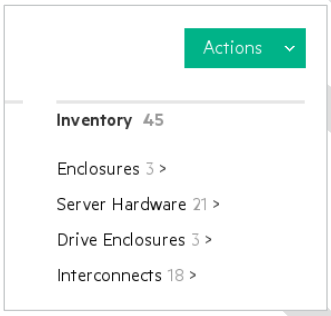
- The OneView dashboard opens with the tutorial. Click **Close**:



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



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



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

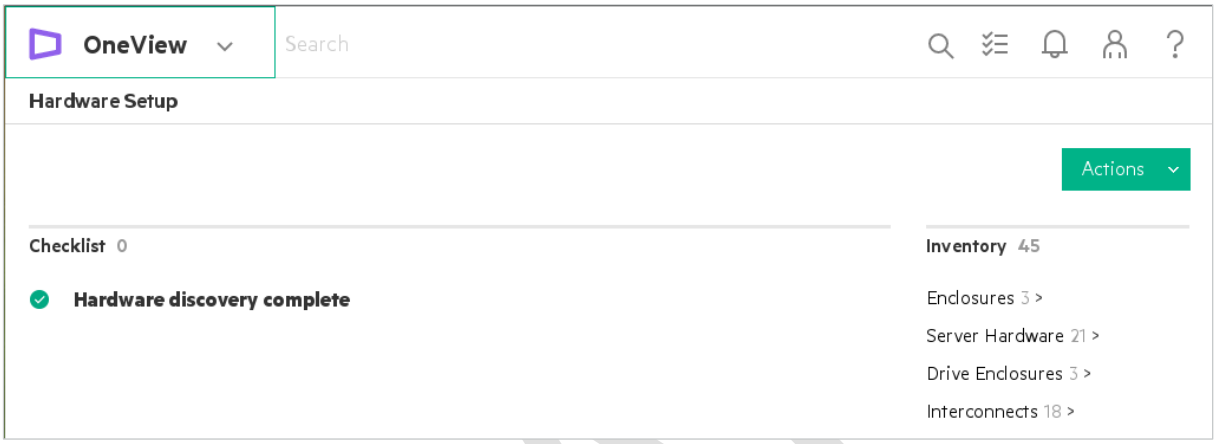
> **Important notice**: It is recommended to wait for the discovery to complete as the lack of CPU/memory resources during this phase can produce negative effects on the appliance. The discovery usually takes about 15-20mn.

Creation of an initial setup snapshot

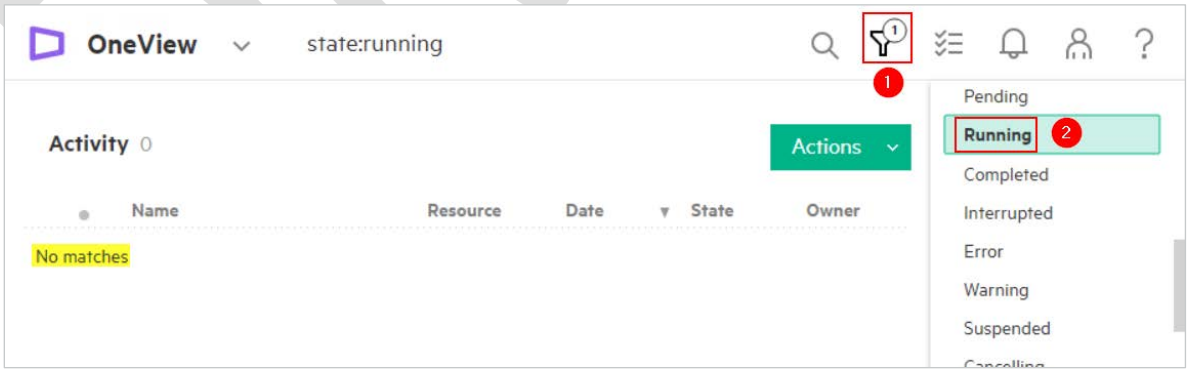
Once the initial appliance configuration is complete, we can create a first snapshot so that we have the initial configuration saved.

Before creating the snapshot:

- Make sure the initial hardware discovery is complete



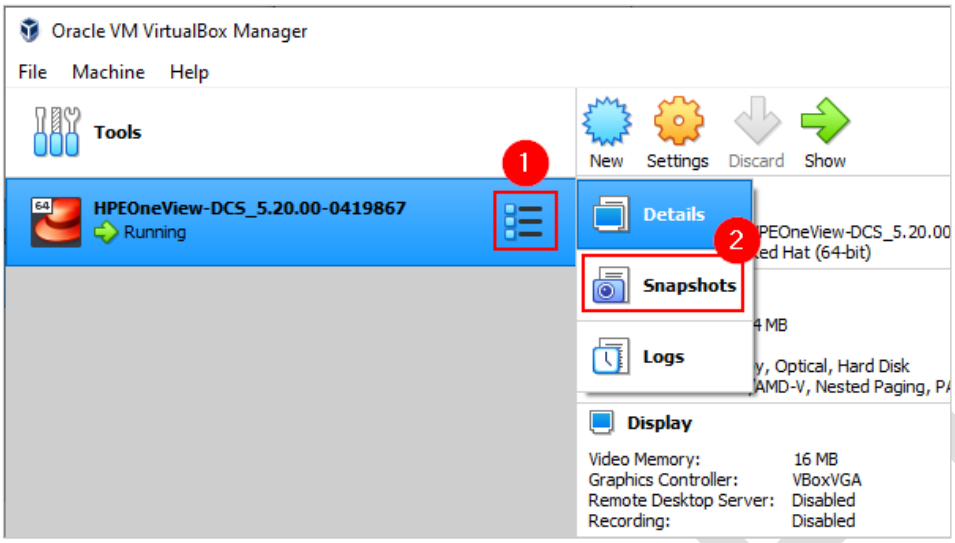
- It is a good practice to wait 5-6 minutes to give time for the environment to stabilize
- Try to resolve any issues that may be reported by the appliance
- Make sure there is no OneView tasks that is still running
    - Go in OneView / **Activity** then use **Running** in the Filter tool:



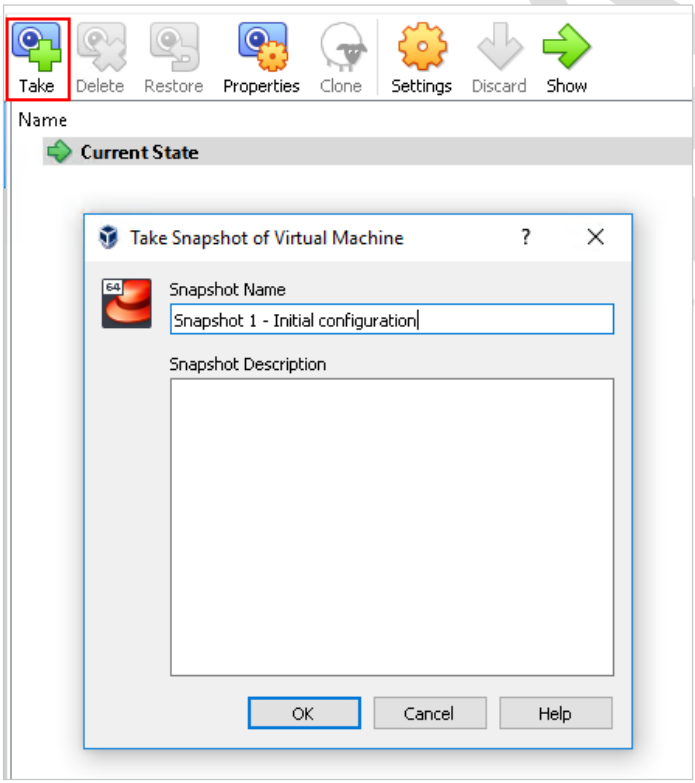
> **Note**: Taking a snapshot while the appliance is running is a key practice to get the appliance up and running almost instantaneously.

To create a snapshot with the initial configuration:

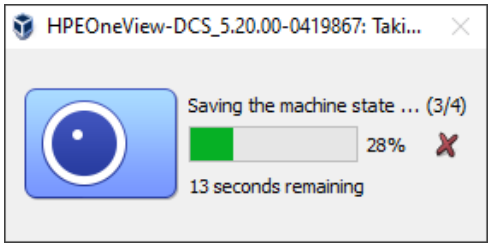
- In VirtualBox, select the **Tools** menu



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



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



This concludes Chapter-4

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

Chapter-5 – Preparing Postman

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

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

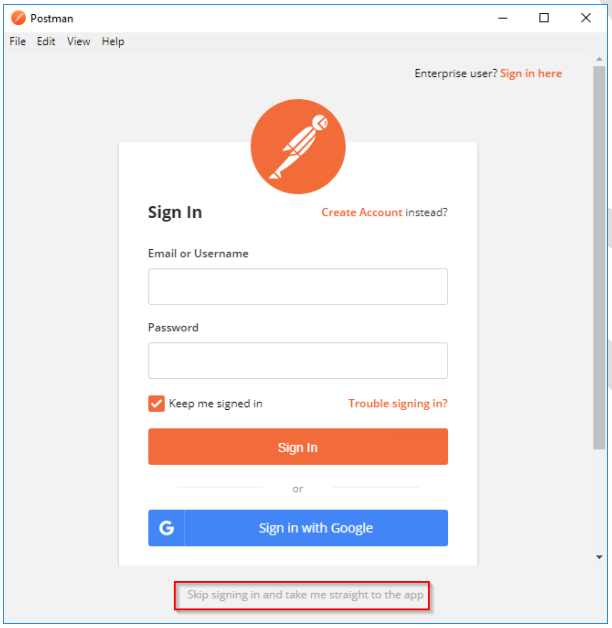
There are several simple solutions for this. My favorite is Postman as it is one of the most complete REST tools and offers useful features:

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

Installing and configuring Postman

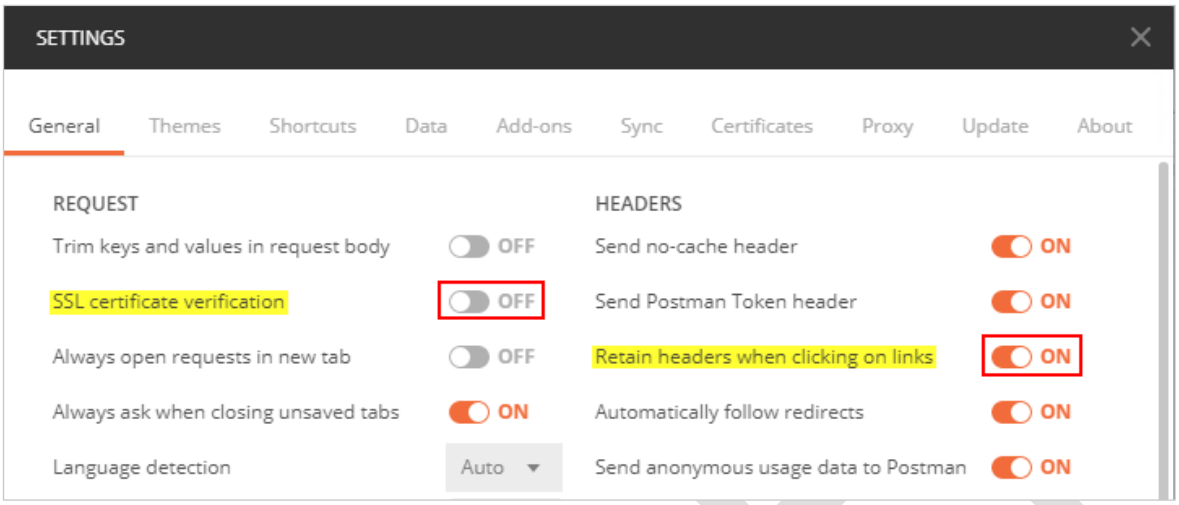
Postman can be downloaded from https://dl.pstmn.io/download/latest/win?arch=64

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



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

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



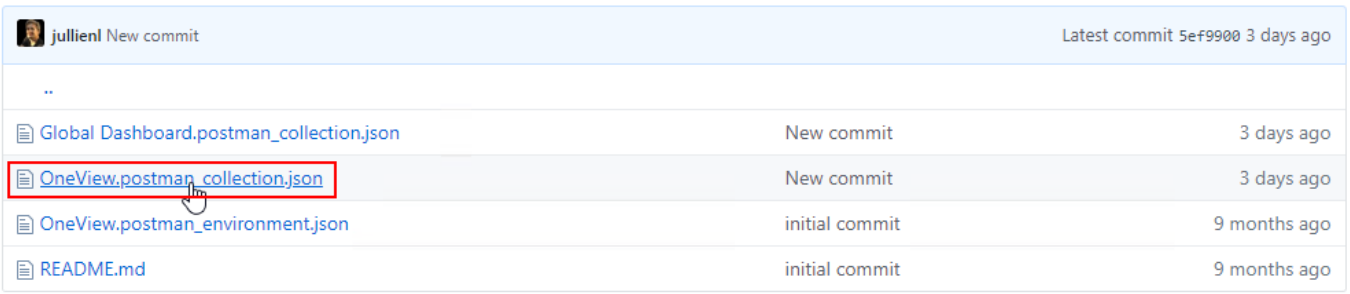
- Set also **Retain headers when clicking on links** to **ON** for greater convenience when clicking on links.

Importing collections

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

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

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



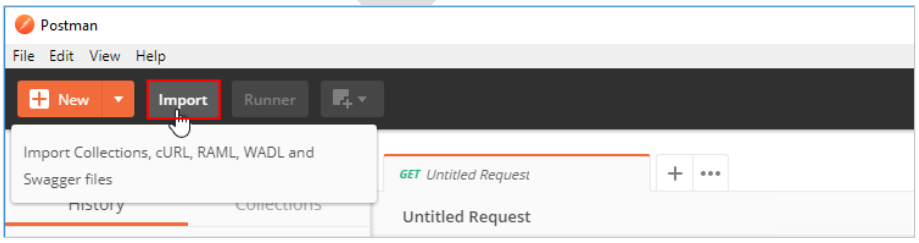
Note: Right-click then **Save link** *as* to save the collection on your system will give you a format not recognized error message when importing in Postman.

- Then click **Raw**

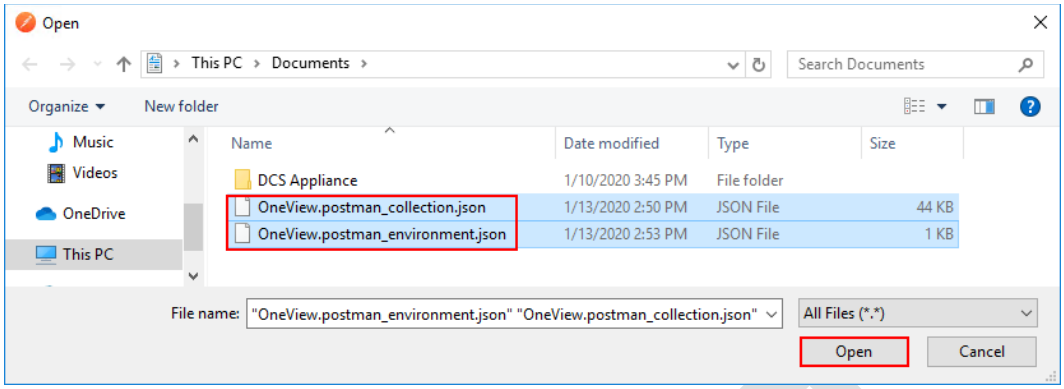
- Then right-click the page and click on **Save as...** to save the JSON file on your system:

- Do the same for **OneView.postman_environment.json**.

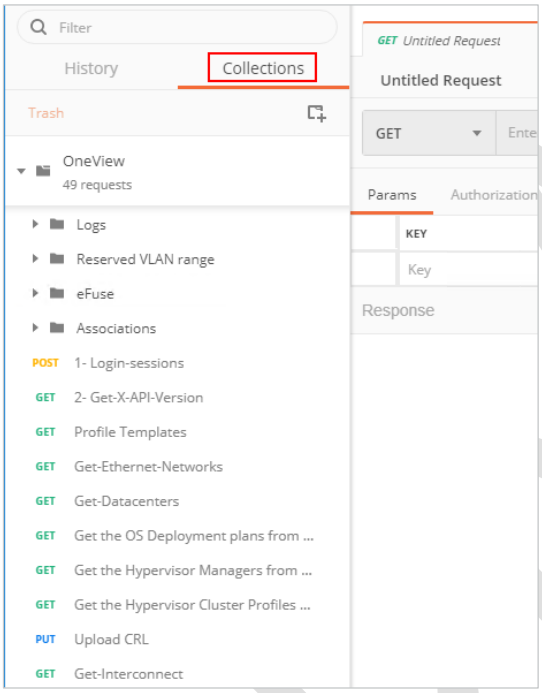
- Back in Postman, click on **Import**:



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



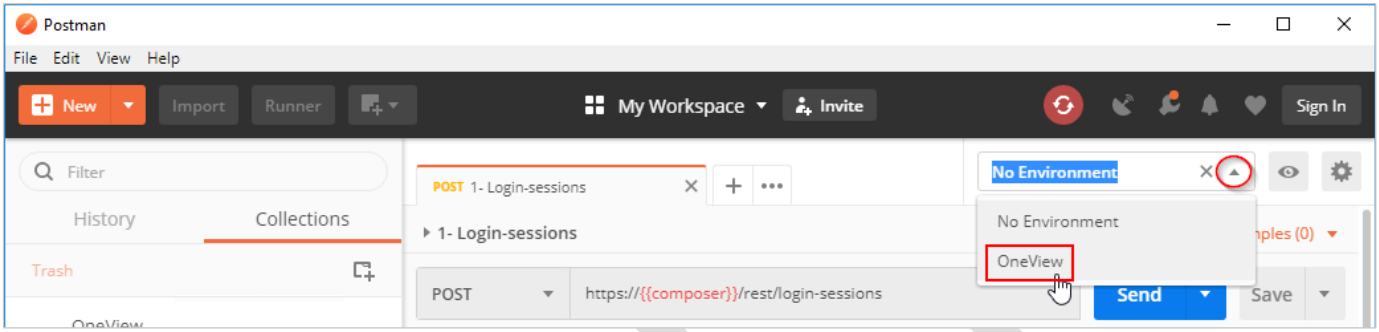
- You should see in collections, the new *OneView* collection



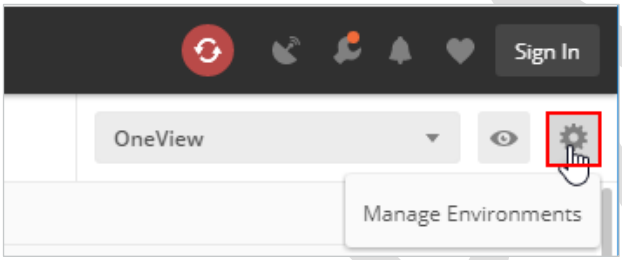
Configuration of the Postman environment

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

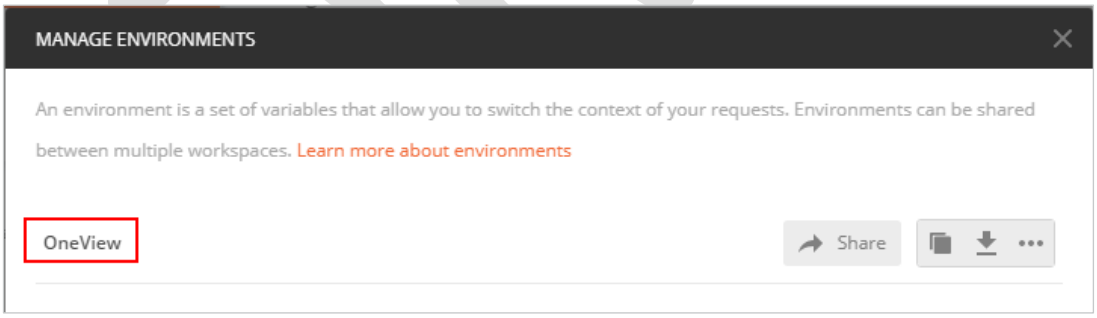
- Expand the environment menu and select **OneView**



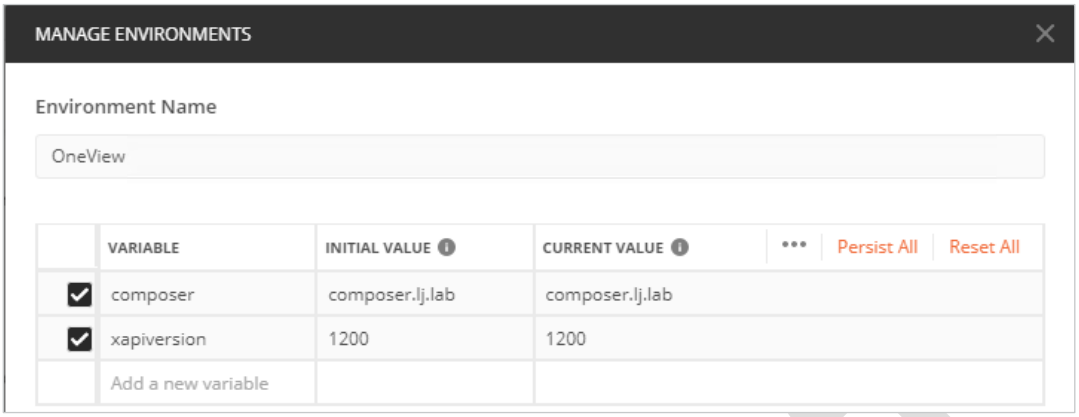
- Then select **Manage Environments**



- Click on **OneView**

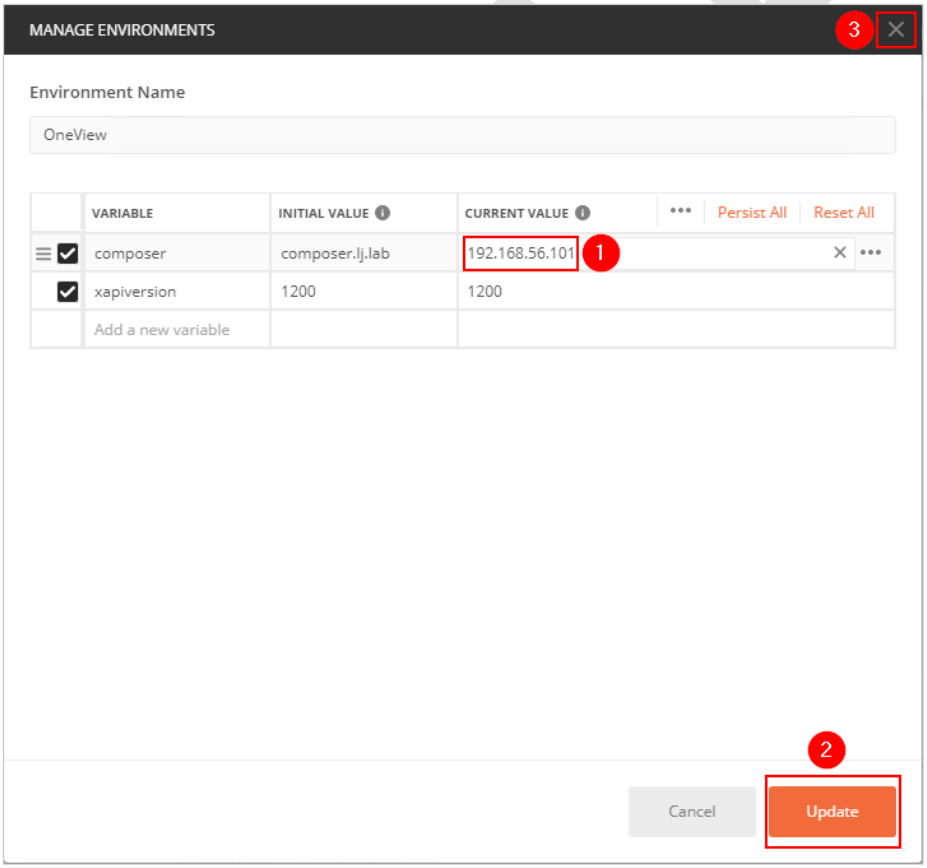


There are two variables defined as illustrated below:

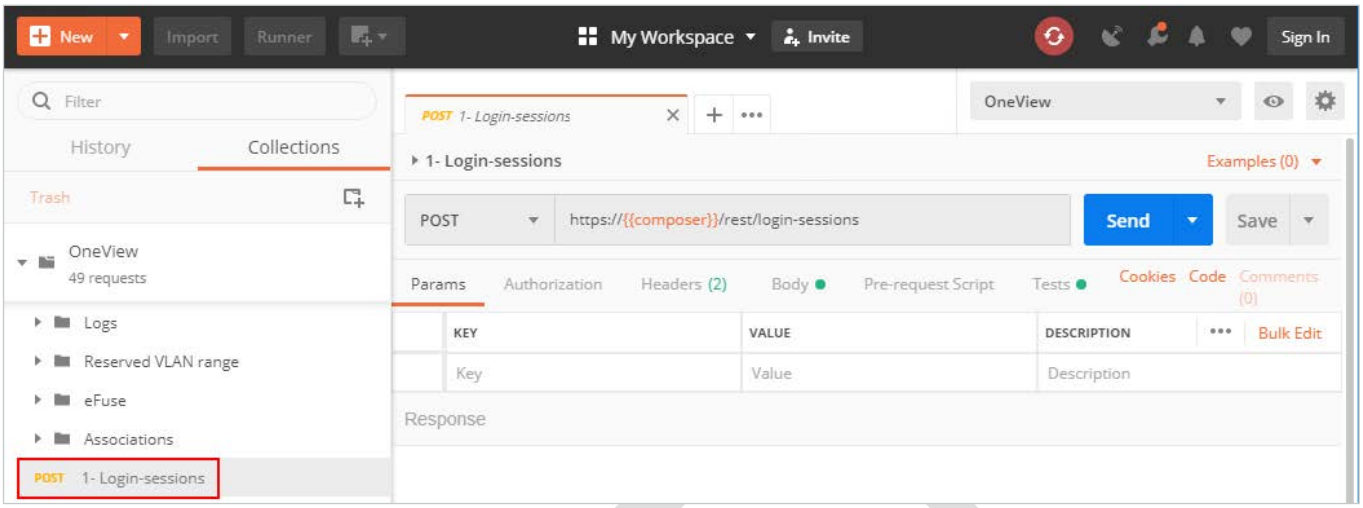


We need to modify the *composer* variable to match with our configuration.

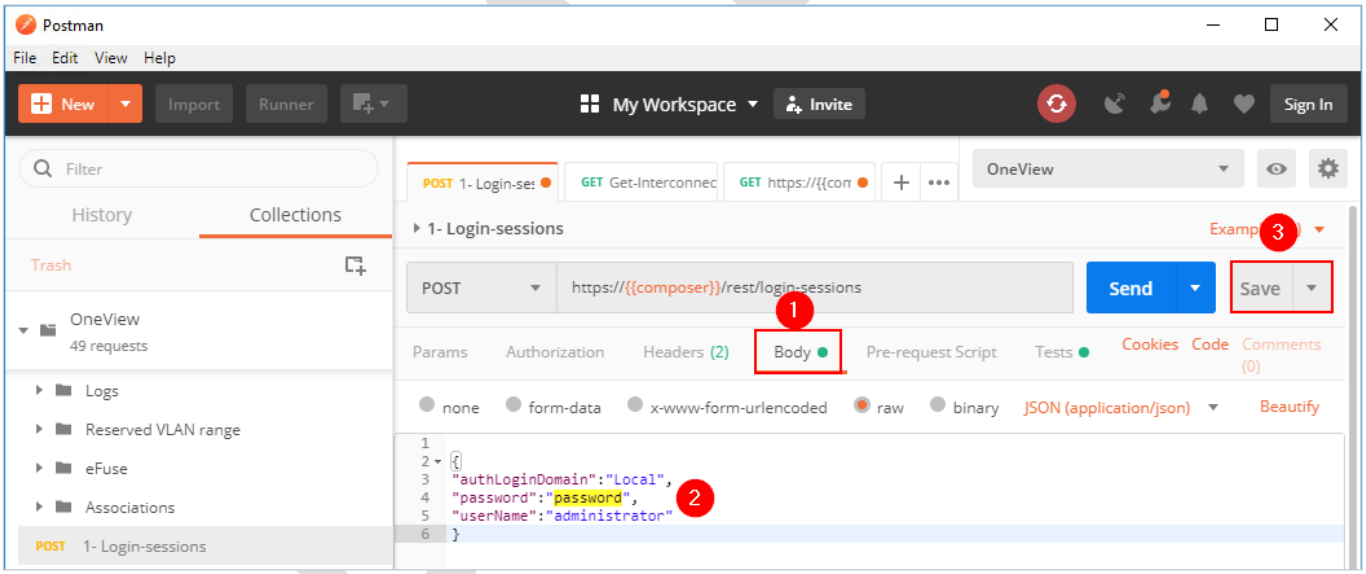
- Change the *composer* value with **192.168.56.101**, the IP address of the appliance then press **Update** then **Close**



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

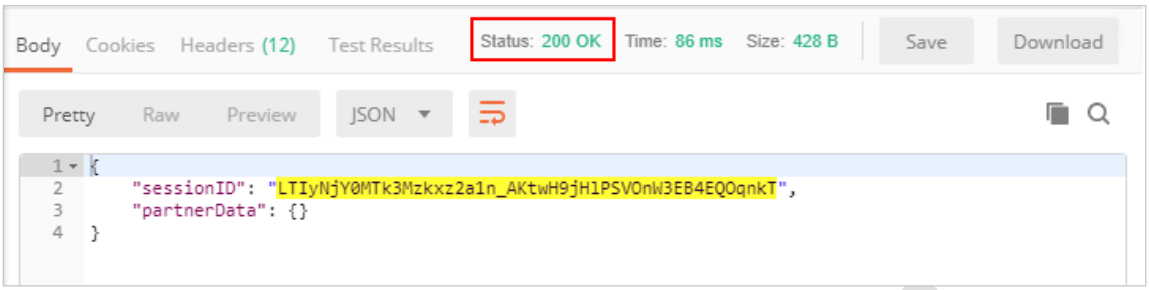


- Then we need to modify the password used by Administrator, select **Body** then enter the password you set in the previous section then click **Save**



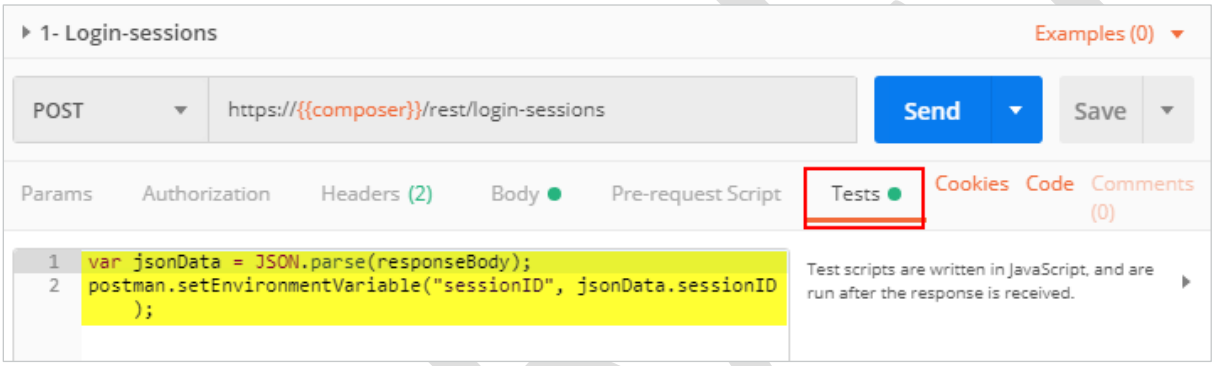
- Then press **Send**

The response we should get is the following:

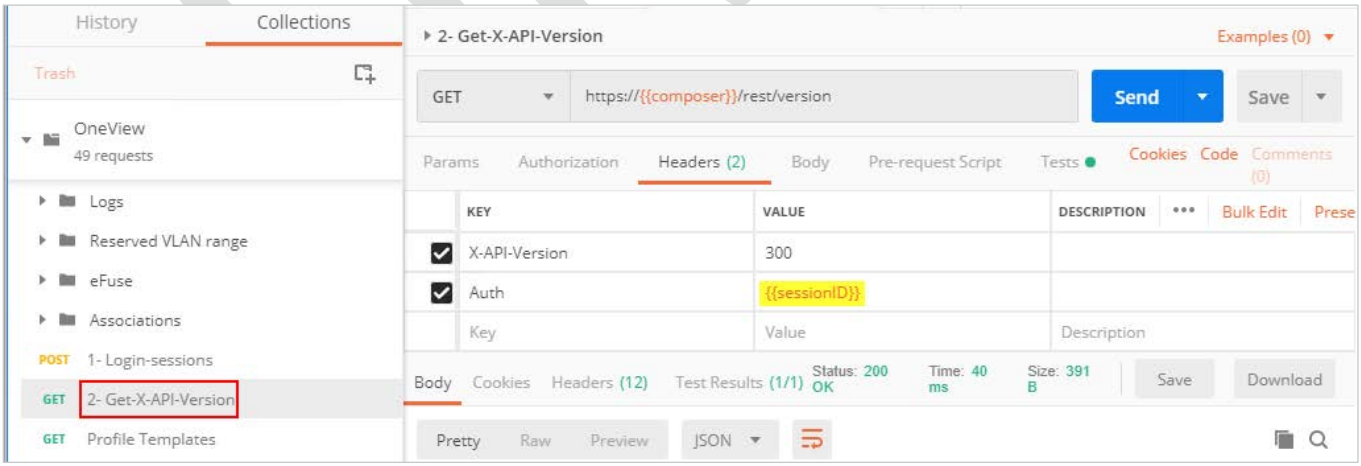


In the Response section, as illustrated above, you want to check the HTTP status code. A value of `200` means it was successful. In the body section, you should get a session ID for the authentication.

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



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.

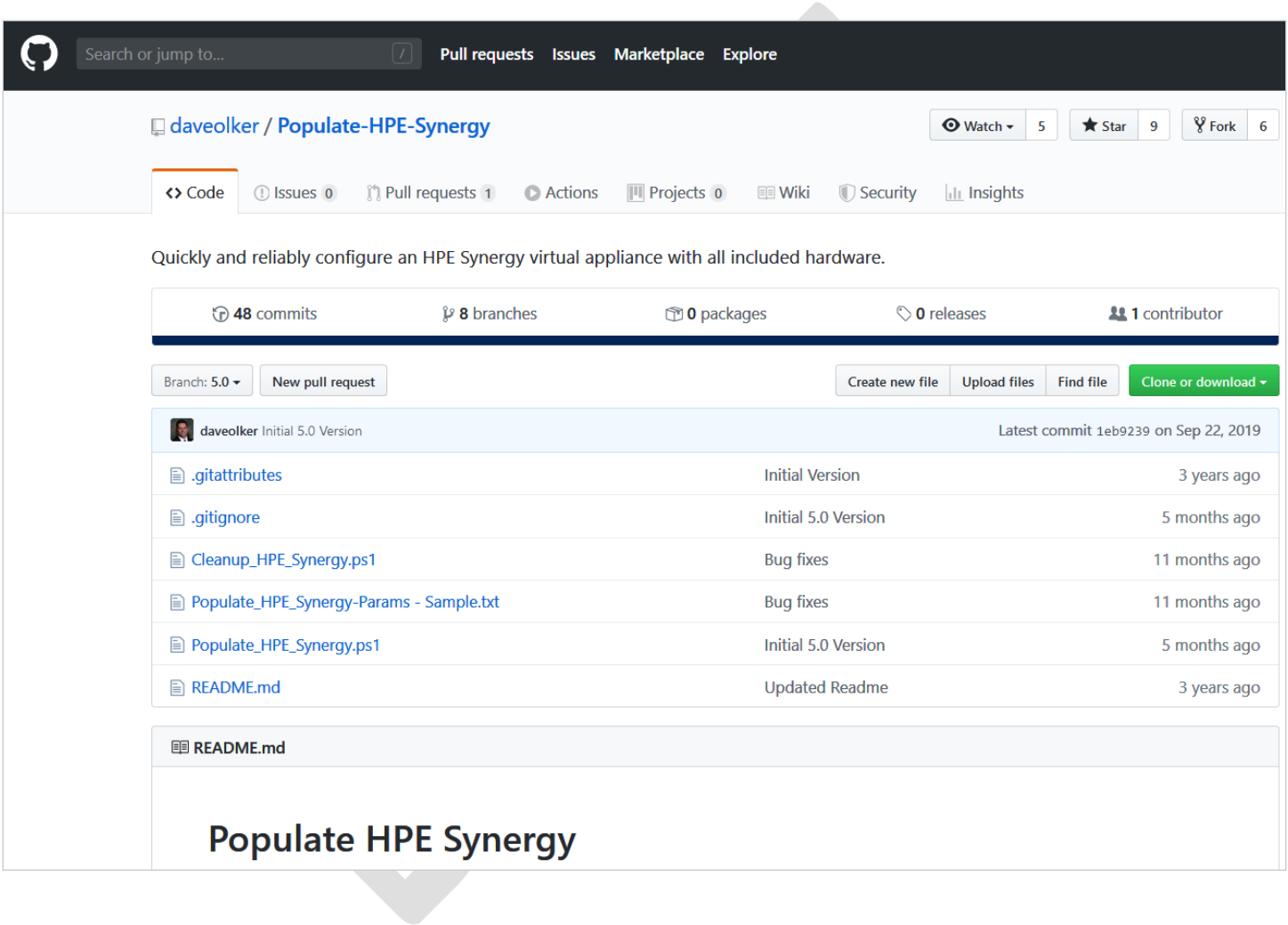
   ⇨ Can be used to run demos with an already configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

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.

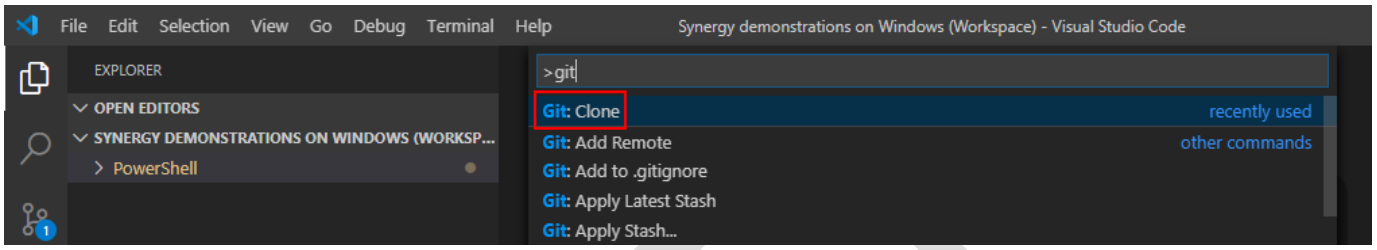
Dave Olker from HPE maintains a GitHub repository (https://github.com/daveolker/Populate-HPE-Synergy-DCS) with a powerful script to configure and populate entirely the HPE OneView DCS demonstration appliance.



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.

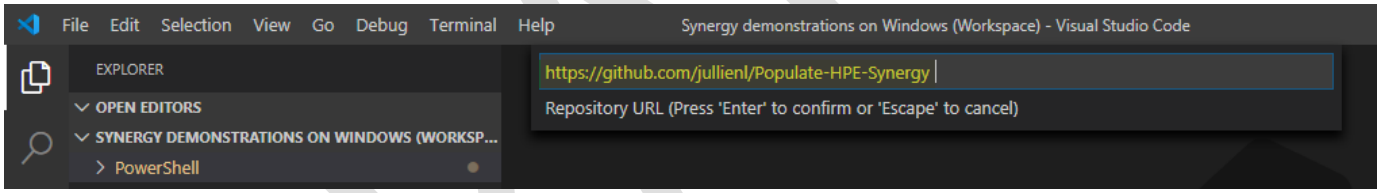
To clone the Dave's repo:

- 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:

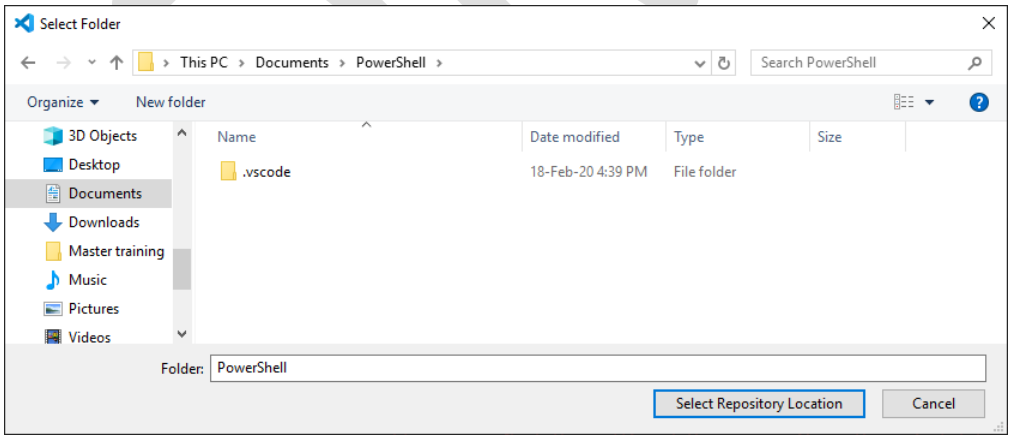


- Enter the following link: **https://github.com/jullienl/Populate-HPE-Synergy**

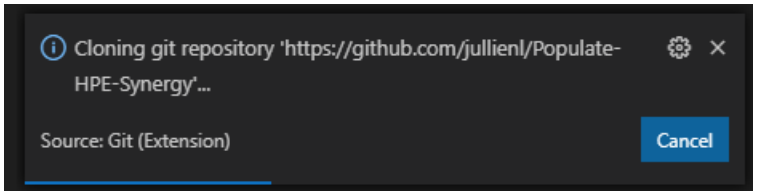
> **Note:** This repository is forked from daveolker/Populate-HPE-Synergy. This fork provides a few changes from Dave' repository to meet the needs of our demonstration scenarios.



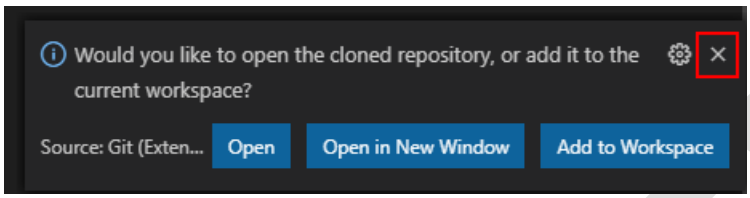
- Then select your Windows user home workspace folder as the parent directory under which to put the Dave's forked repository:



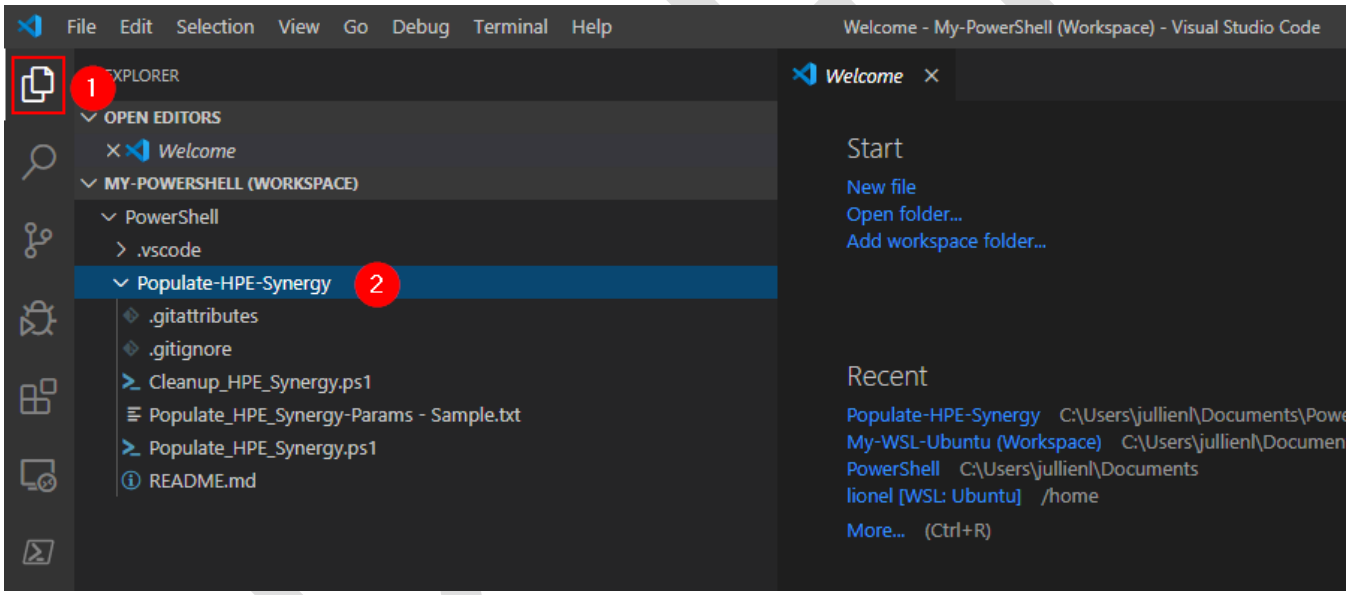
- Once selected, the cloning starts:



- Just close the pop-up window for now



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

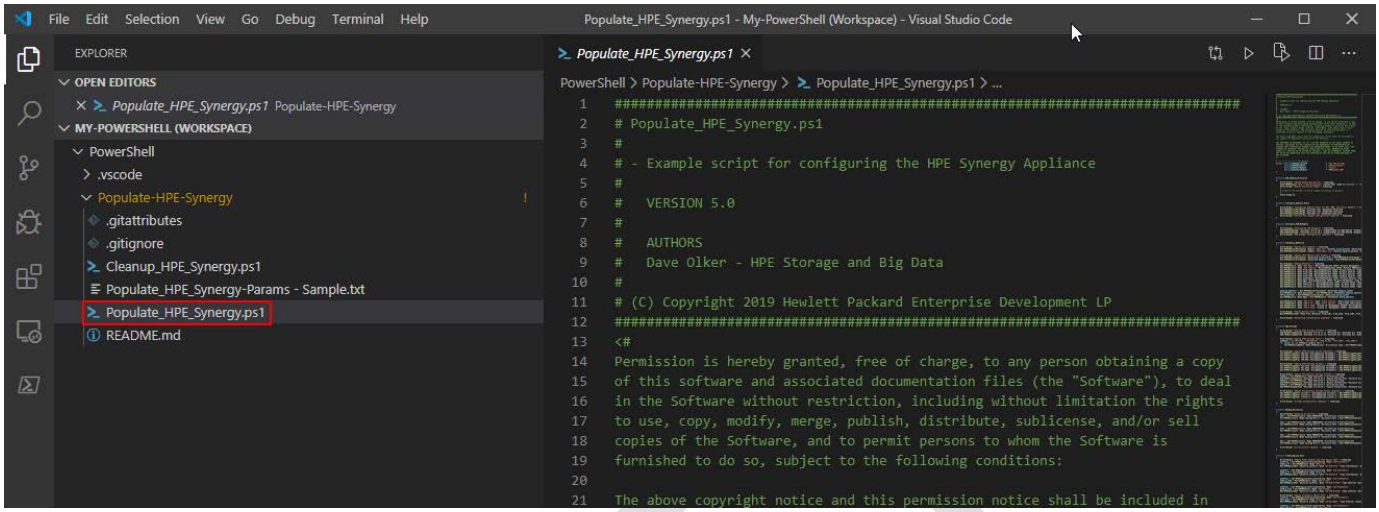


This repository provides two scripts, one to configure and populate entirely the HPE OneView DCS demonstration appliance and one to clean up everything:

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

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

- Open **Populate_HPE_Synergy.ps1**



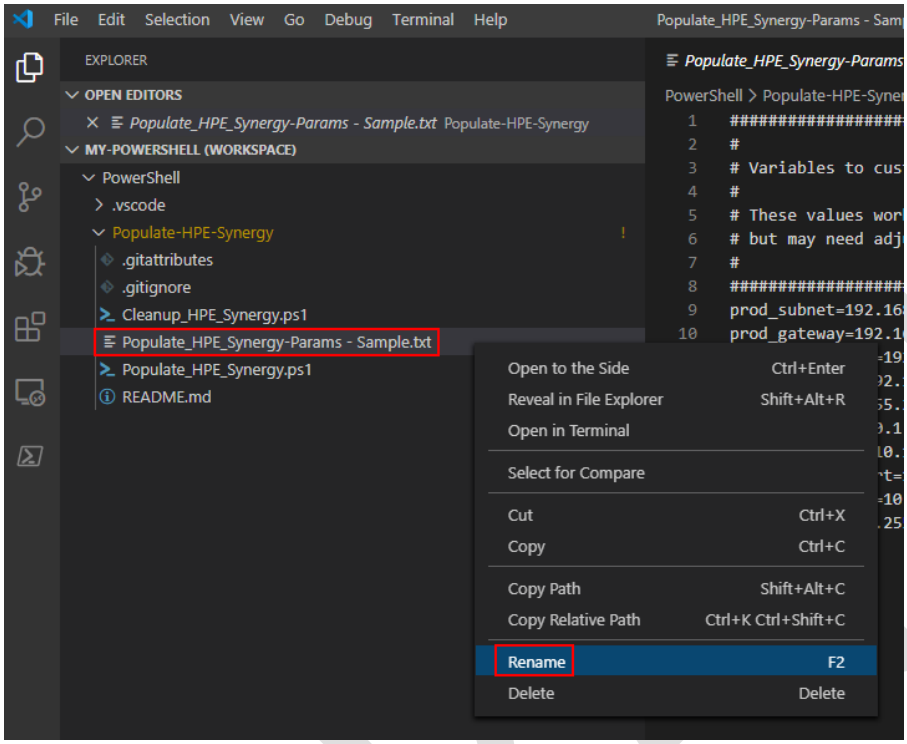
As described in the README.md, this script does the following:

- 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

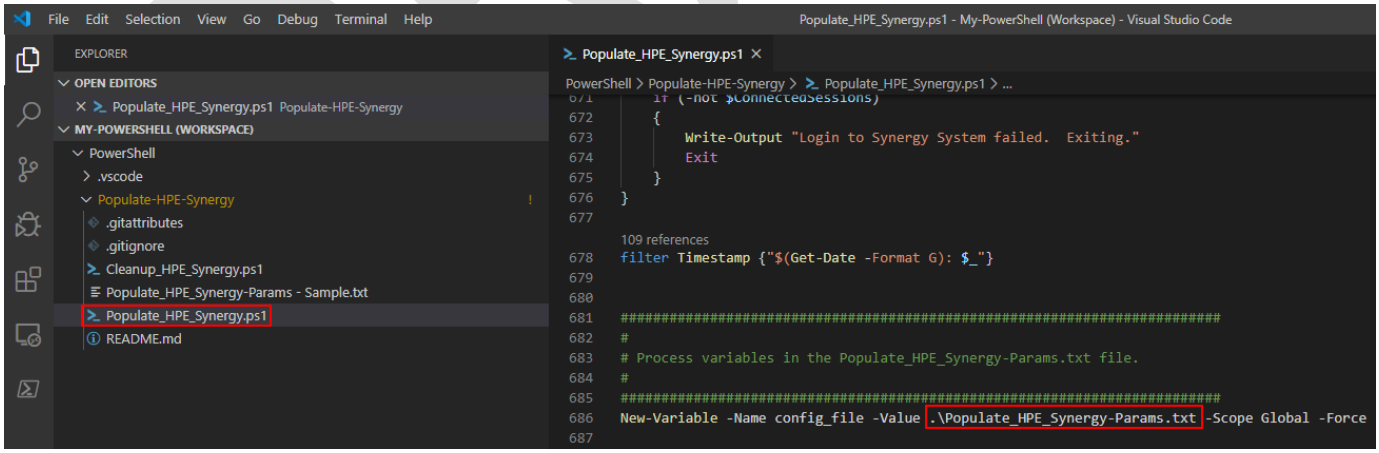
> **Note**: To get a better display of the script, press **CTRL** + **K** + **CTRL** + **0** to collapse all regions

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

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



This configuration file is defined line 686 in the script:



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

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

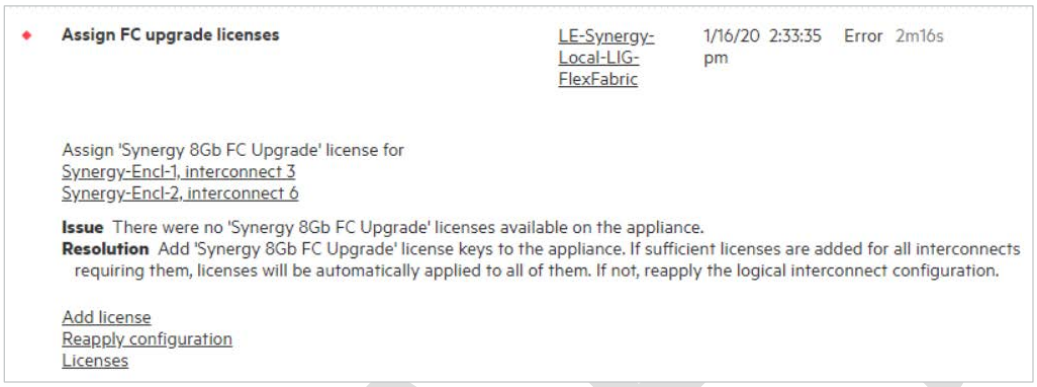
Each function runs a specific task:

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

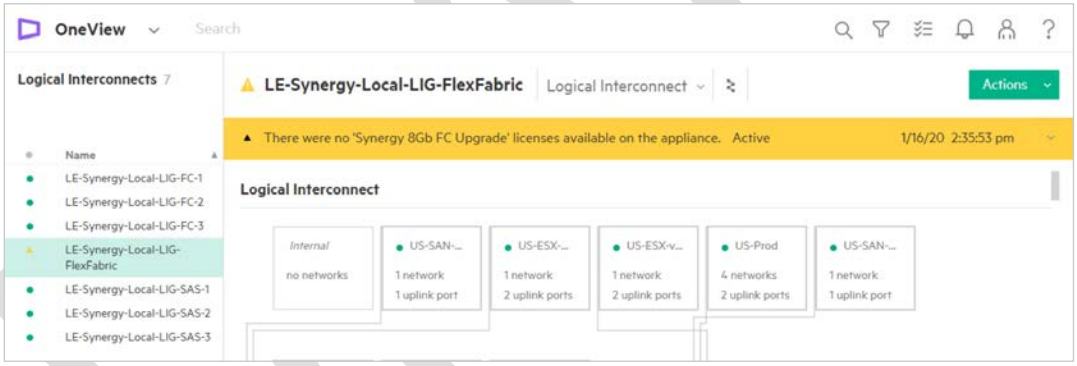
> **Note**: You can download an HPE Synergy SPP from https://www.hpe.com/downloads/synergy
>
> **Note**: You can always upload it later to your OneView appliance by going to Firmware Bundles in the Appliance section of the main menu.

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

- `Add_Licenses` adds a Synergy Fibre Channel license to OneView. When run, the function asks to specify the filename containing the license. OneView licenses is not required for a Synergy environment. For Fibre Channel connectivity with the Virtual Connect SE 40 Gb F8 Module, a license is required. Without this license, FC ports cannot be activated but this is not blocking any operations other than:
    - Throwing an error during the creation of the Logical Enclosure:



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



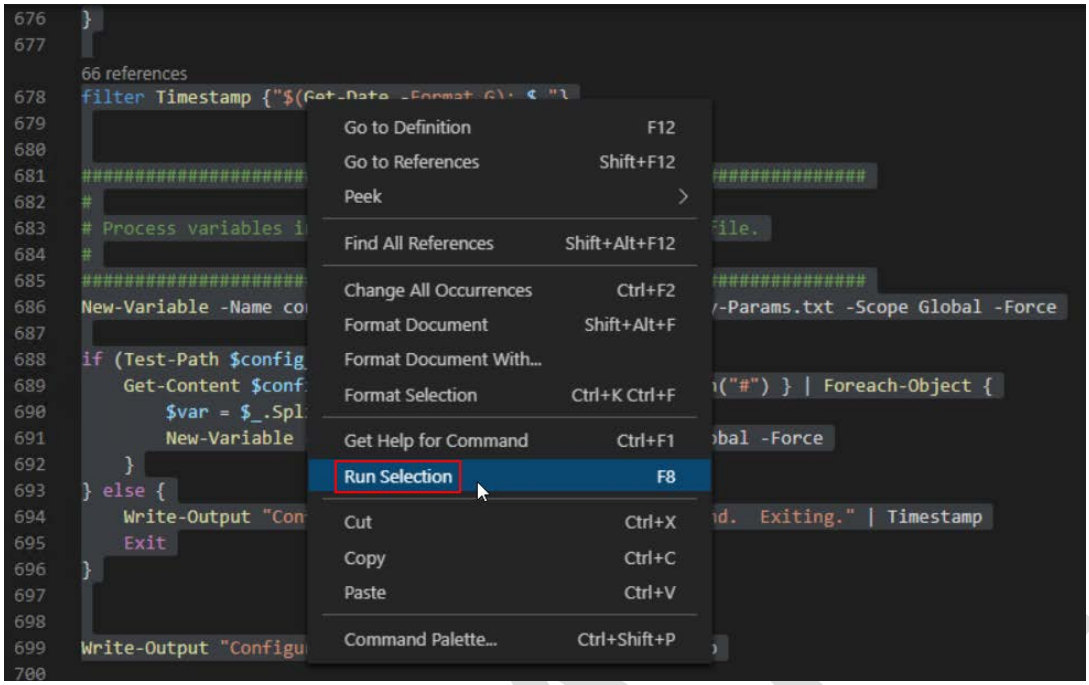
- `Configure_Address_Pools` configures the virtual address pools for MAC, WWN, and Serial Numbers.
- `Add_Remote_Enclosure` adds two additional enclosures in OneView (optional).
- `Rename_Enclosures` renames the 5 enclosures with more convenient names (e.g. Synergy-Encl-1, Synergy-Encl-2, etc.)
- `PowerOff_All_Servers` turns all servers off to prepare the creation of Server Profiles
- `Configure_SAN_Managers` adds two Cisco MDS 9250i switches
- `Configure_Networks` creates 15 networks (Ethernet, FC and FCoE)

- `Add_Storage` adds 2 x 3PAR 7200 Storage Systems and 3 x StoreVirtual then adds 3 x Volumes and 7 x Volume Templates
- `Add_Users` creates 5 new users with different roles.
- `Create_OS_Deployment_Server` configures the enclosures for Image Streamer
- `Create_Logical_Interconnect_Groups` creates 3 x LIGs (SAS, FC and FlexFabric)
- `Create_Uplink_Sets` configures 8 x uplink sets (2xFC, 2xFCoE, 1xMgmt, 1x vMotion, 1xImageStreamer, 1xProd)
- `Create_Enclosure_Group` creates an EG with 3 frames/LIGs + Image Streamer
- `Create_Logical_Enclosure` creates the LE with EG/LIGs/Streamer configured previously
- `Add_Scopes` creates a new scope with the first frame and the production networks
- `Create_Server_Profile_xxx` creates different server profiles types using Gen9 and Gen10 servers

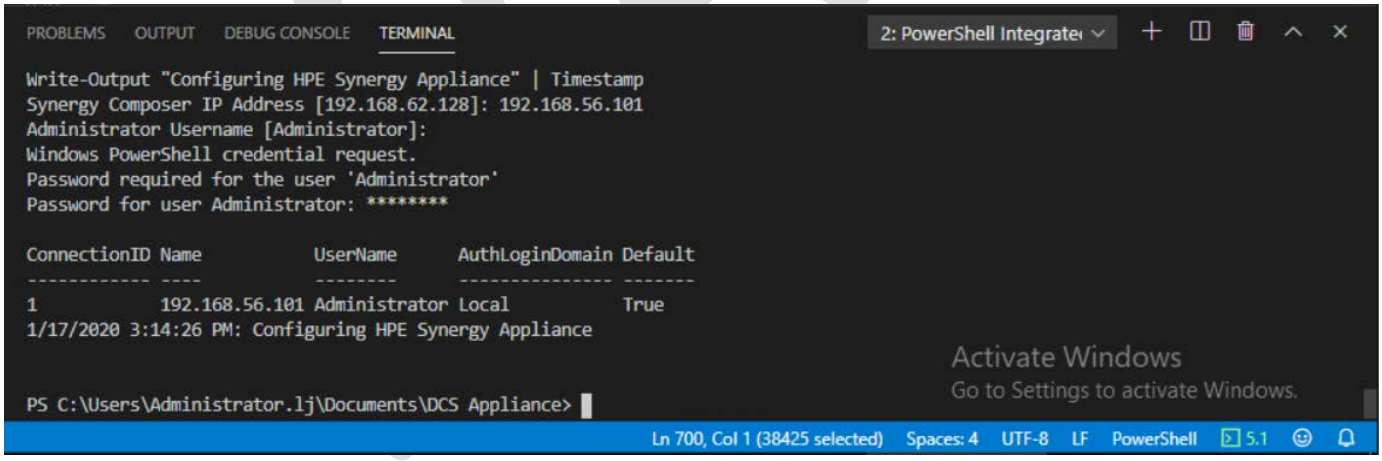
- Now select the first part of the script from line 1 to 699

```
}

66 references
filter Timestamp {"$(Get-Date -Format G): $_"}


##############################################################################
#
# Process variables in the Populate_HPE_Synergy-Params.txt file.
#
##############################################################################
New-Variable -Name config_file -Value .\Populate_HPE_Synergy-Params.txt -Scope Global -Force

if (Test-Path $config_file) {
    Get-Content $config_file | Where-Object { !$_.StartsWith("#") } | Foreach-Object {
        $var = $_.Split('=')
        New-Variable -Name $var[0] -Value $var[1] -Scope Global -Force
    }
} else {
    Write-Output "Configuration file '$config_file' not found.  Exiting." | Timestamp
    Exit
}


Write-Output "Configuring HPE Synergy Appliance" | Timestamp

Add_Firmware_Bundle
Add_Licenses
Configure_Address_Pools
Add_Remote_Enclosures
Rename_Enclosures
```

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

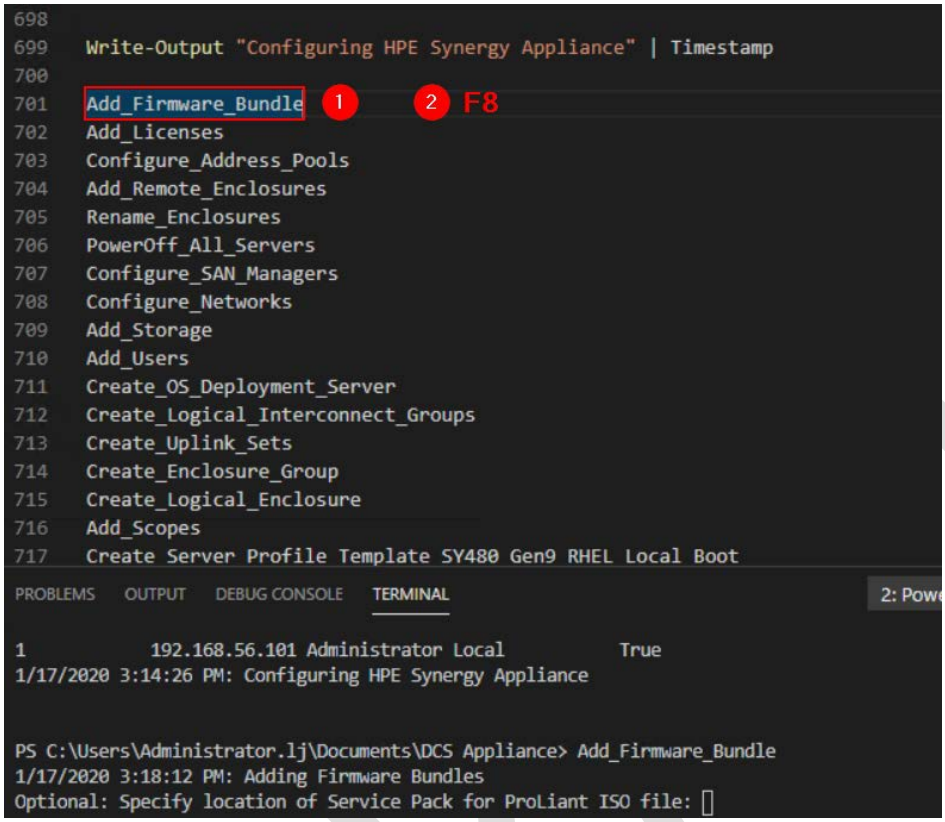


- As requested, enter the Composer IP: **192.168.56.101**
- Then **Administrator** / **password** for the OneView credentials



- Make sure no error is thrown.

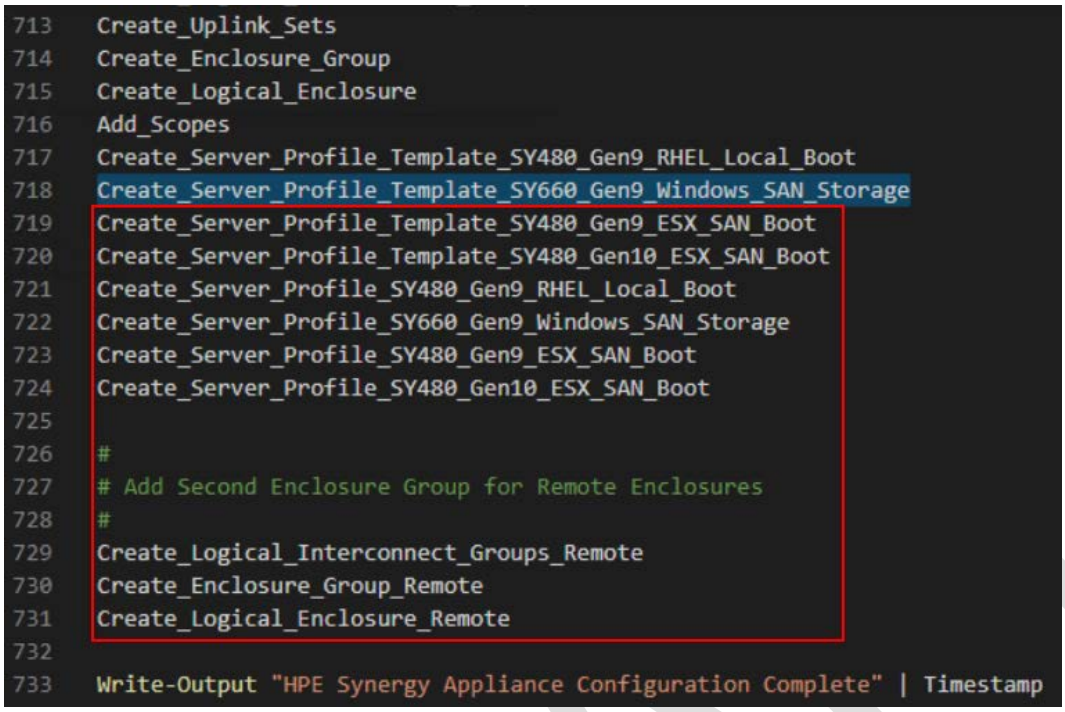
- Then the idea is to select and run one at a time each function by pressing **F8** so that we see the output in the console. You can move back and forth between VS Code and OneView web interface to see the result of each step.



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

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

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



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

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

```
Disconnect-HPOVMgmt
```

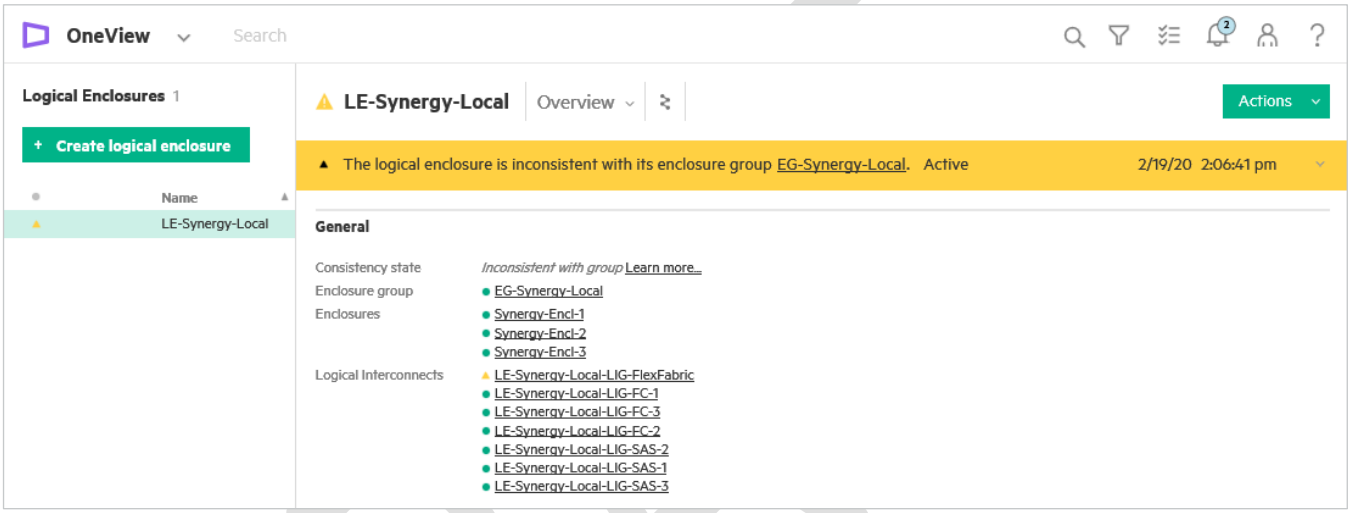
Creation of a final setup snapshot

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

Before creating the second snapshot:

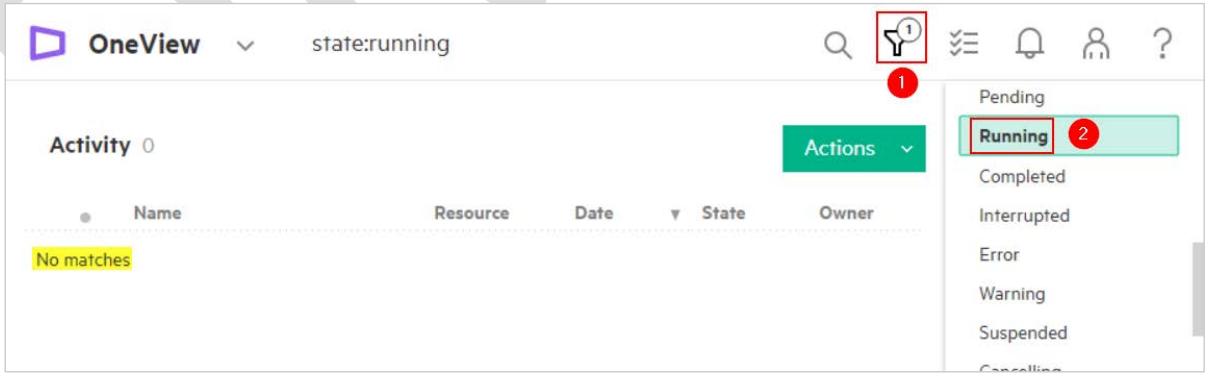
- Try to resolve any issues that may be reported by the appliance

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



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

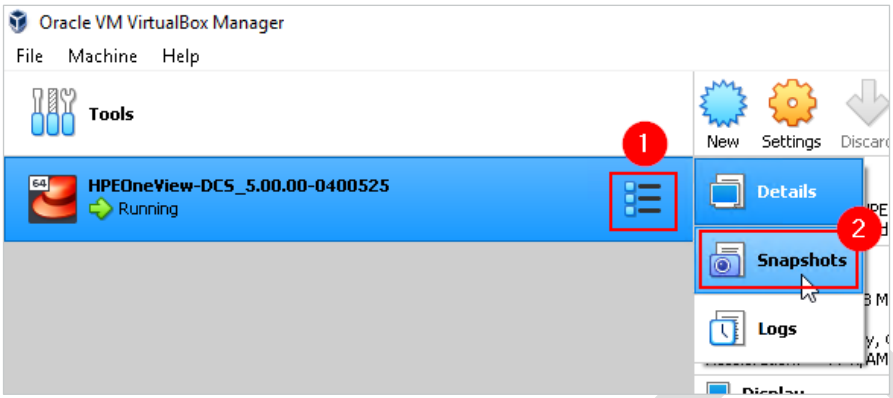
  1. Go in OneView / **Activity** then use **Running** State in the Filter tool:



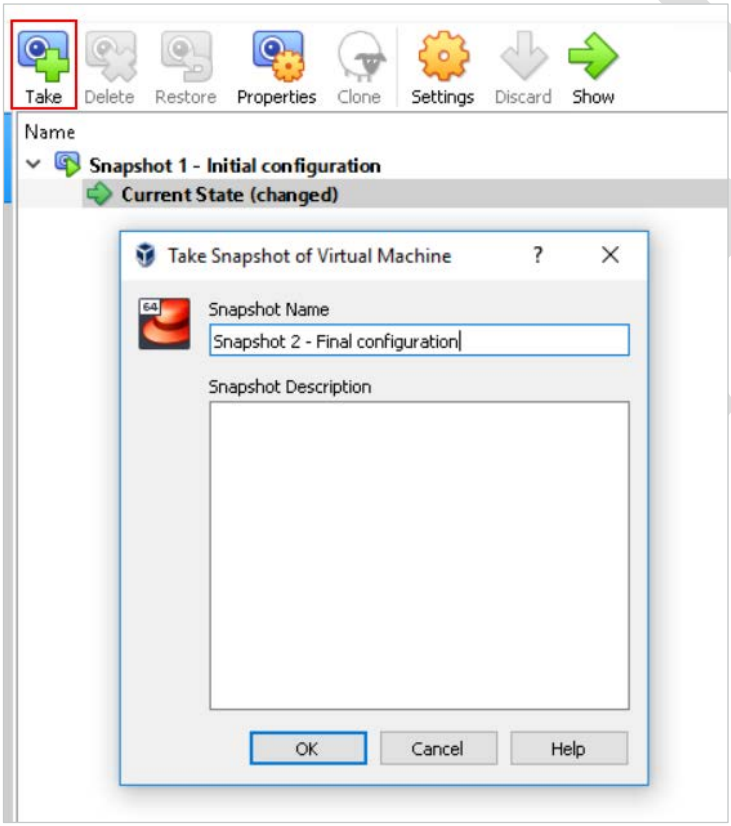
> **Note**: Taking a snapshot while the appliance is running is a key practice to avoid waiting long minutes for the appliance to start.

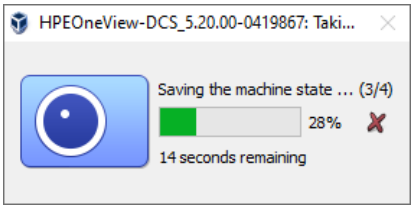
To create a snapshot to save the appliance final configuration:

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

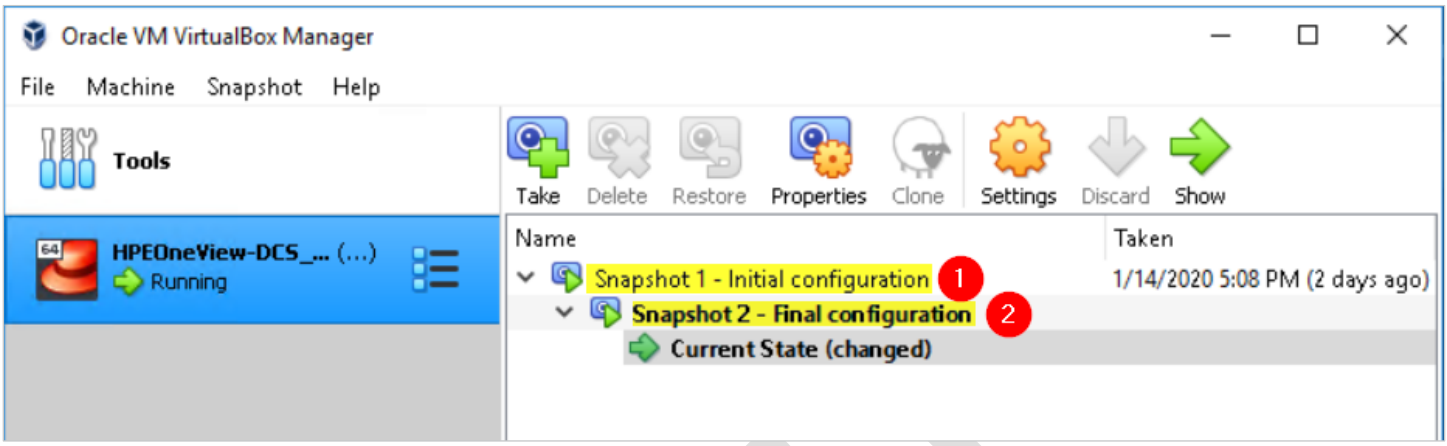


- Then press on **Take**, enter a snapshot name like **Snapshot 2 – Final configuration** then click **OK**





To sum up, we have now two snapshots:



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

2- Second snapshot is with a fully configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

Chapter-7 – Preparing the live demonstration scenarios

Demonstrating Synergy Composer and OneView Key features

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

Demonstrating Infrastructure programmability

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

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

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

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

For Python, we have two scenarios:

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

For Ansible, we have three scenarios:

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

For Terraform, we have three scenarios:

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

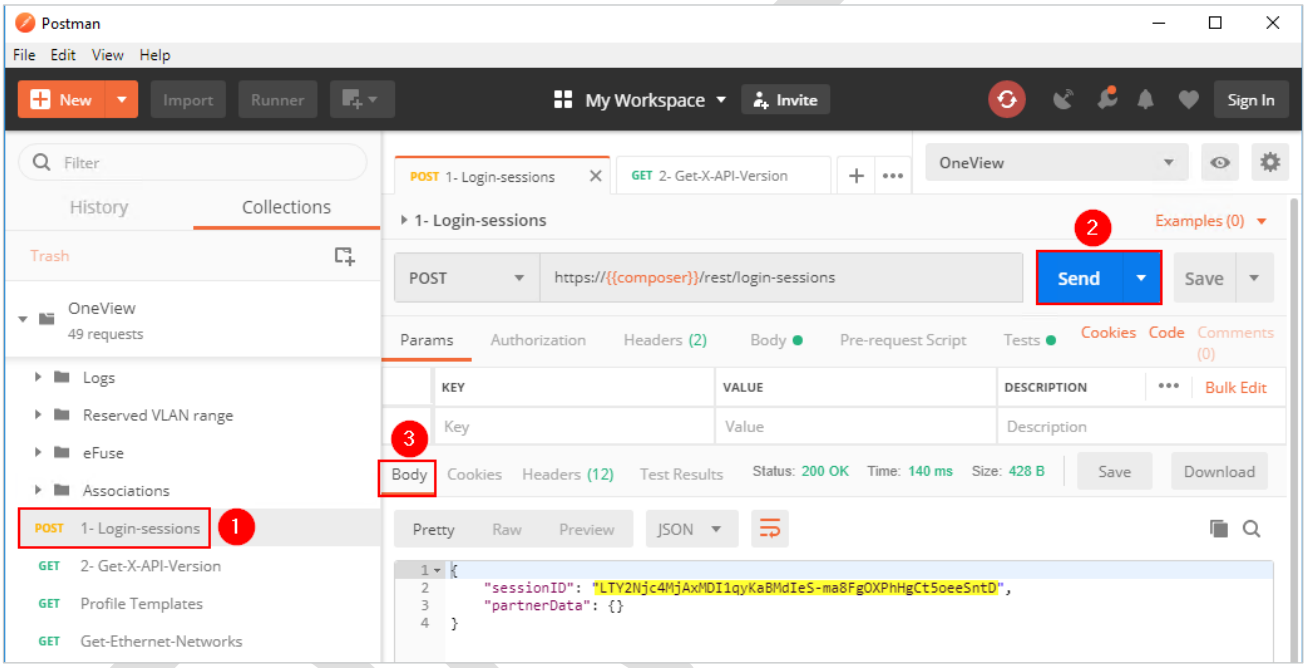
More scenarios will be added over time…

Postman - Scenario 1 – Introduction to the OneView REST API

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

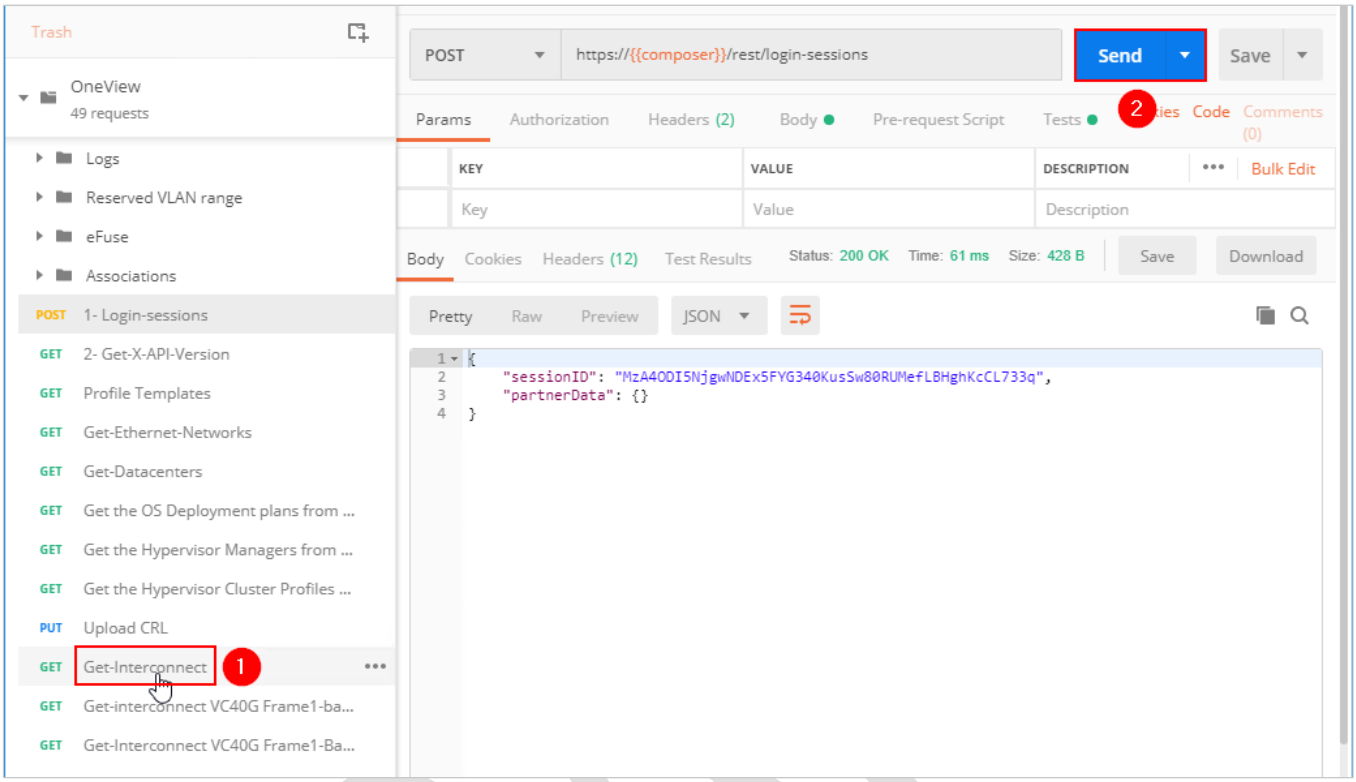
> **Notice**: This scenario can be run with either the initial or final configuration snapshot.

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



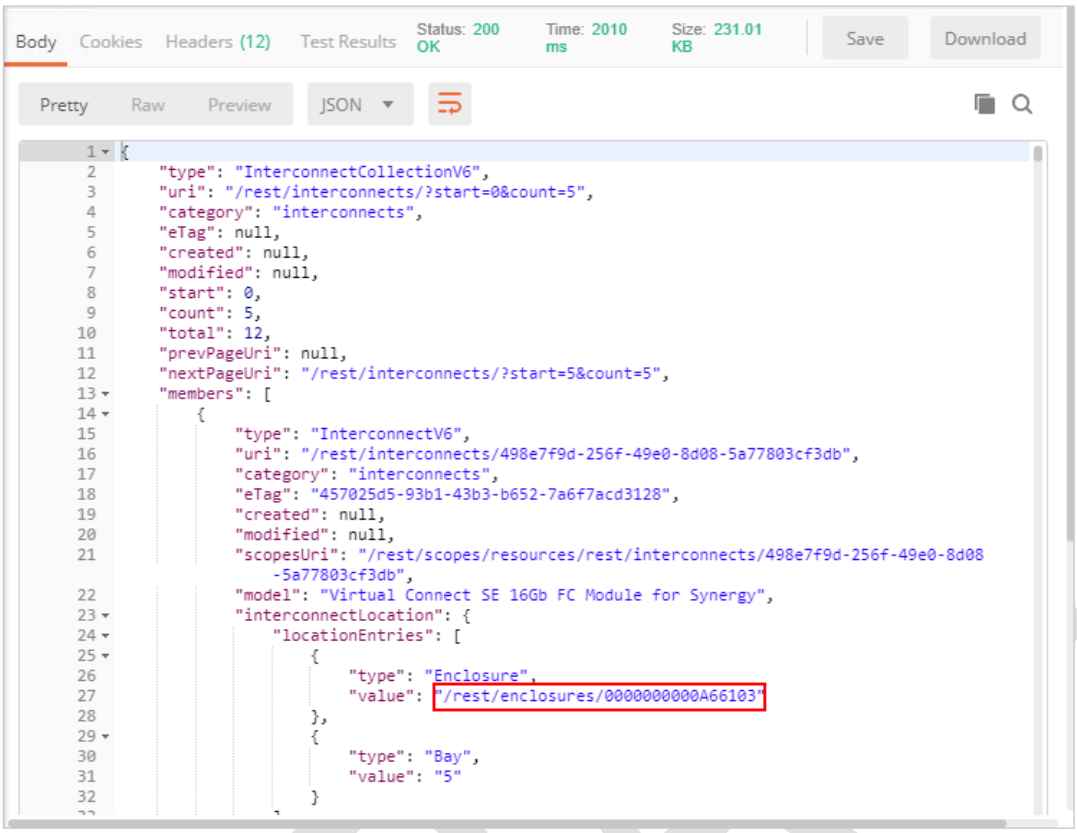
- Explain `sessionID` is our API authentication token that will be used to pass all other REST calls.

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

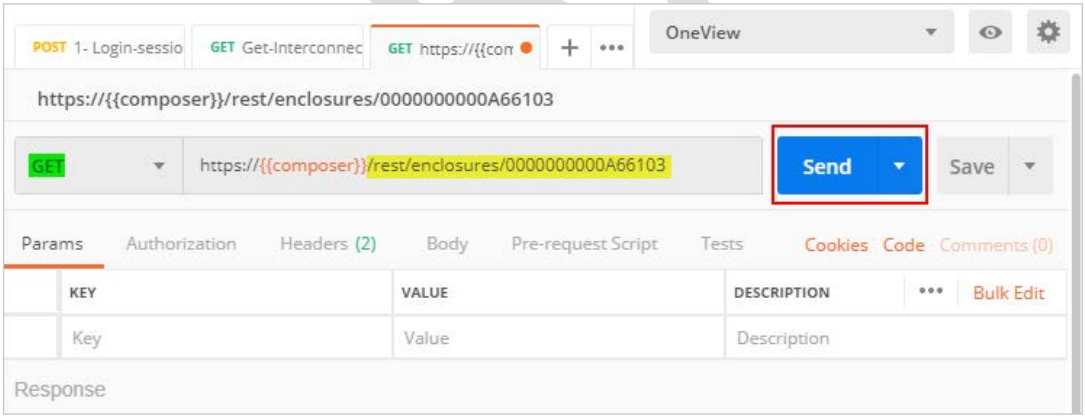


- The response displays all interconnects managed by OneView

- Click on the enclosure link of the first interconnect

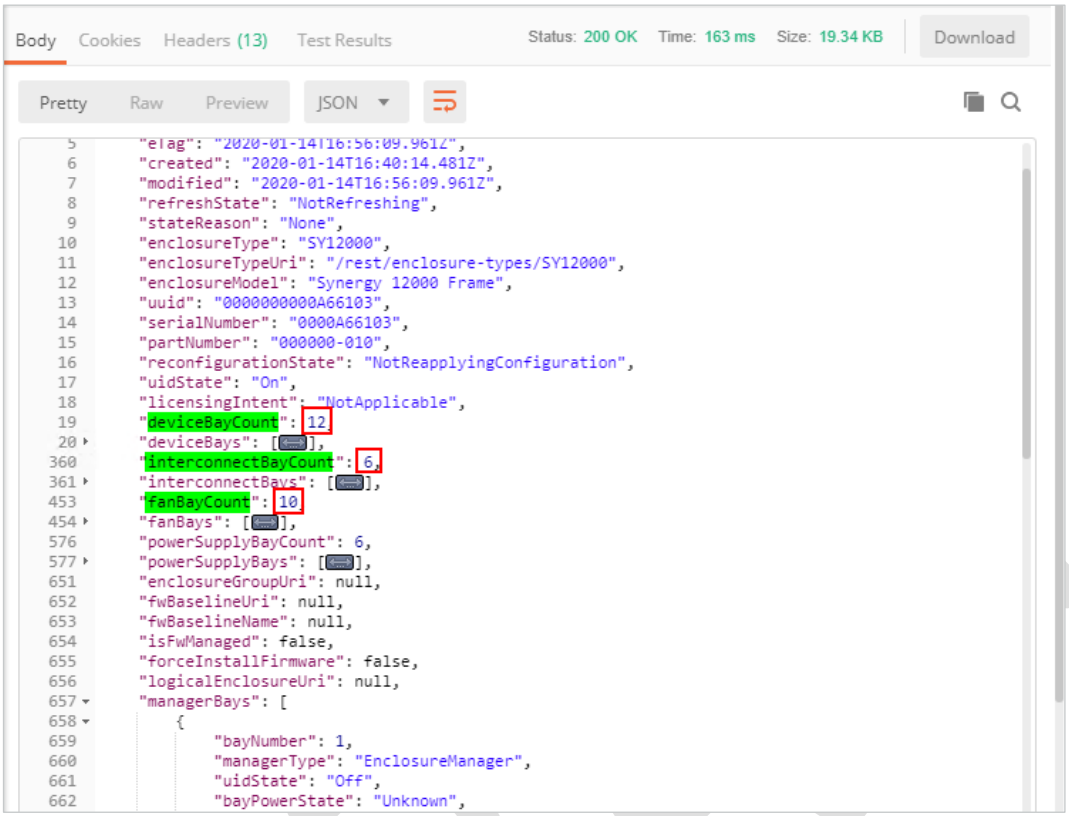


- Notice that when doing so, Postman creates a **GET** request using this URI, press **Send**



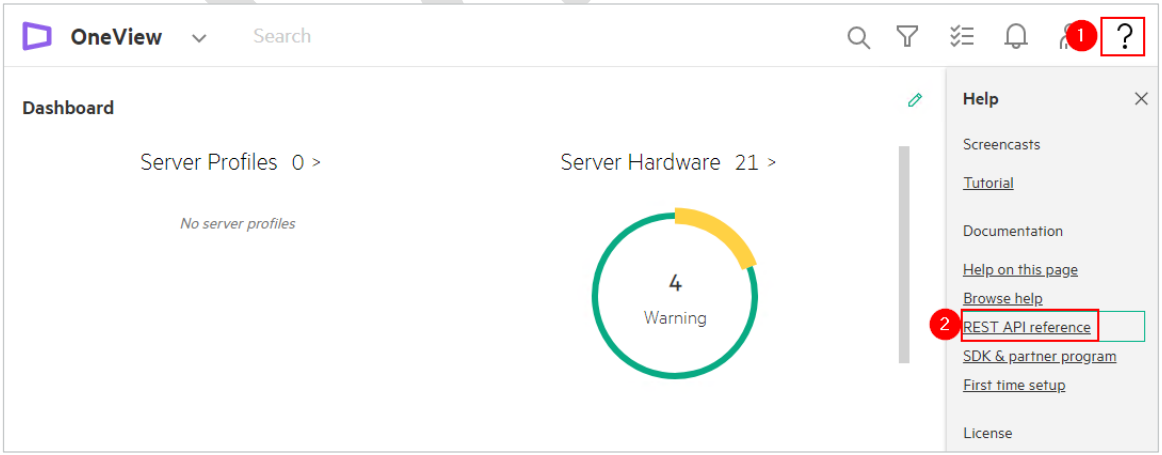
- The response provides information about the enclosure UUID `0000000000A66103`

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

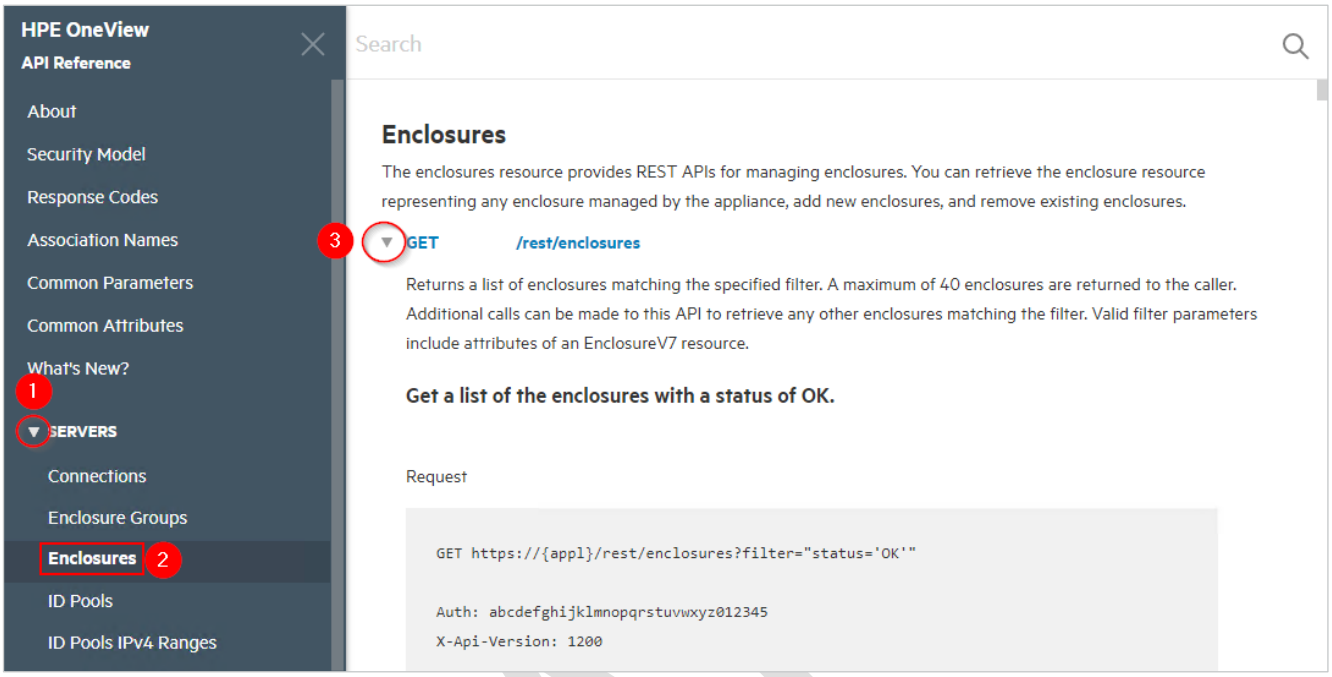


Note: You can use the expand icon to unfold the different sections to get a better display

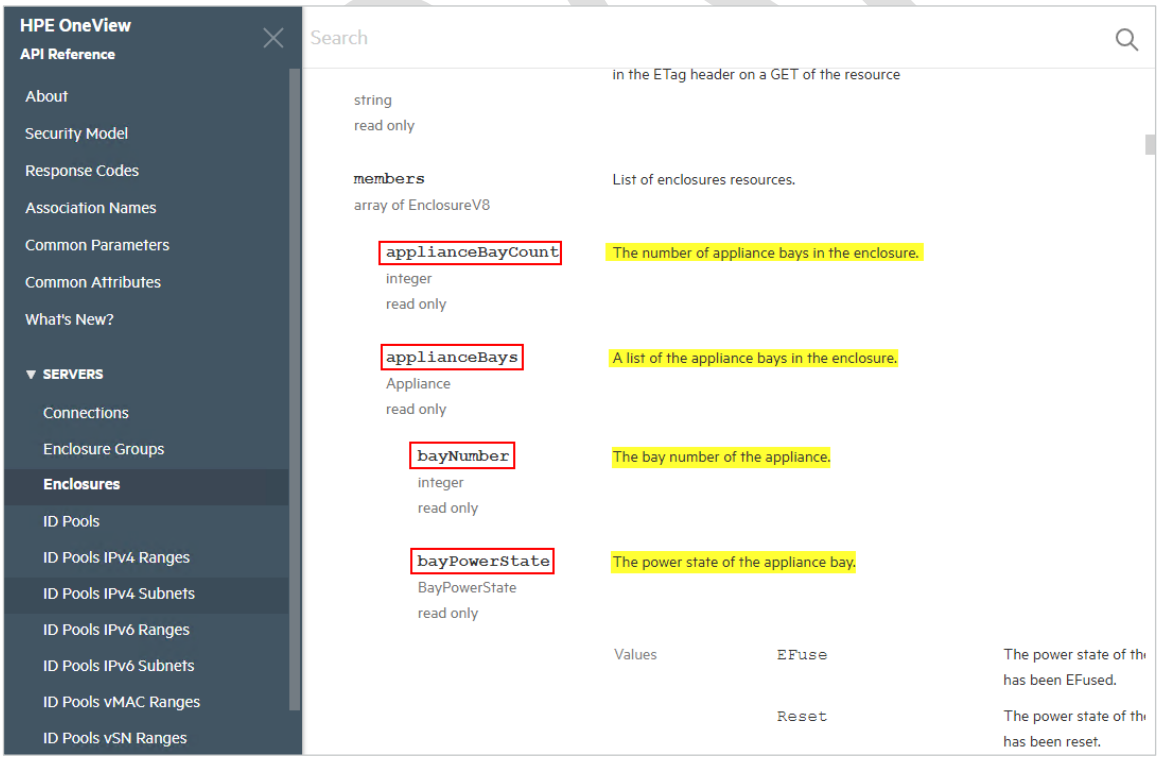
- A description of all these components is available in the REST API Reference document accessible from the Help section of OneView



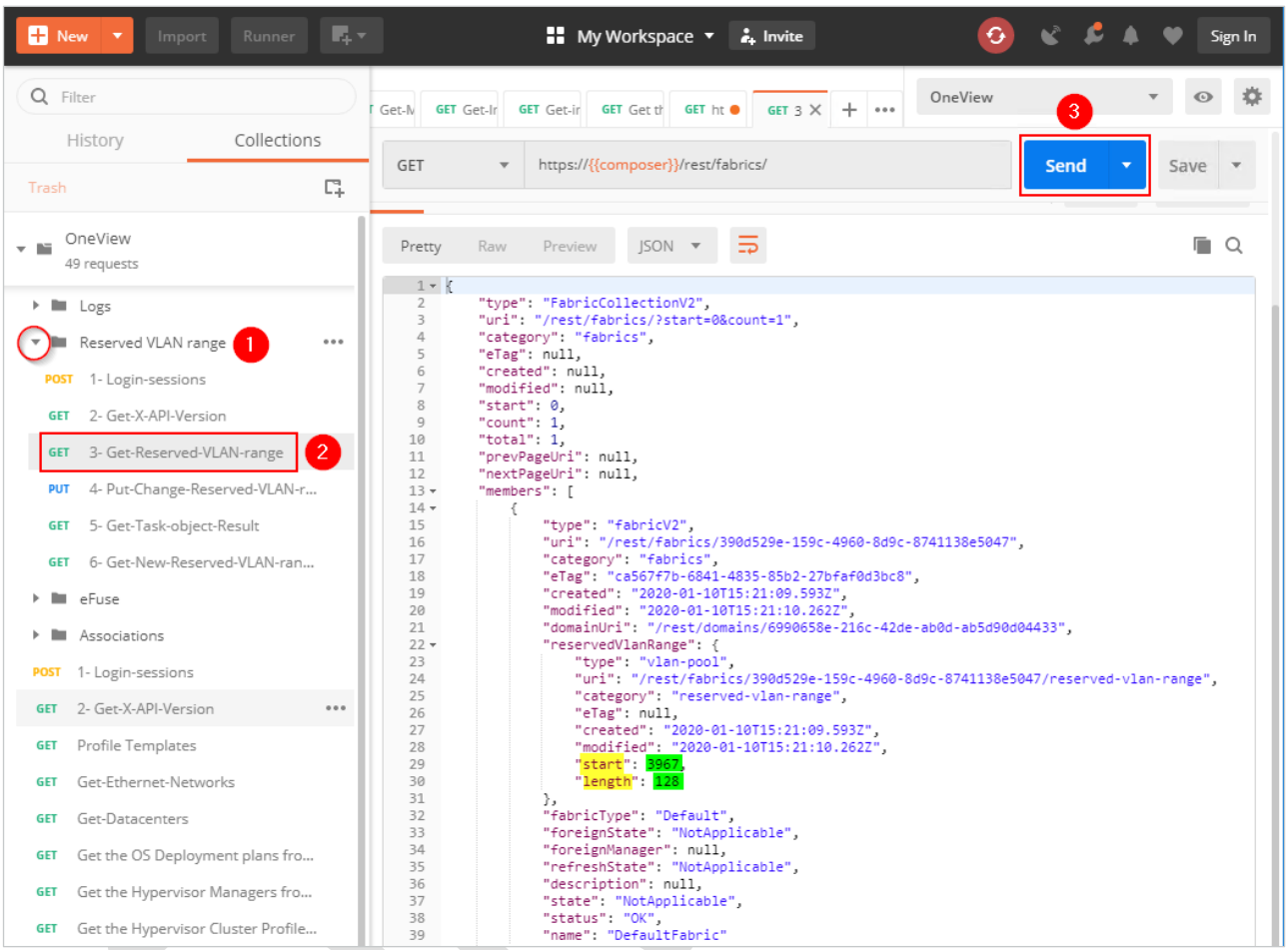
- In the REST API Reference document, select **Server** / **Enclosure** / **Get /rest/enclosures**



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



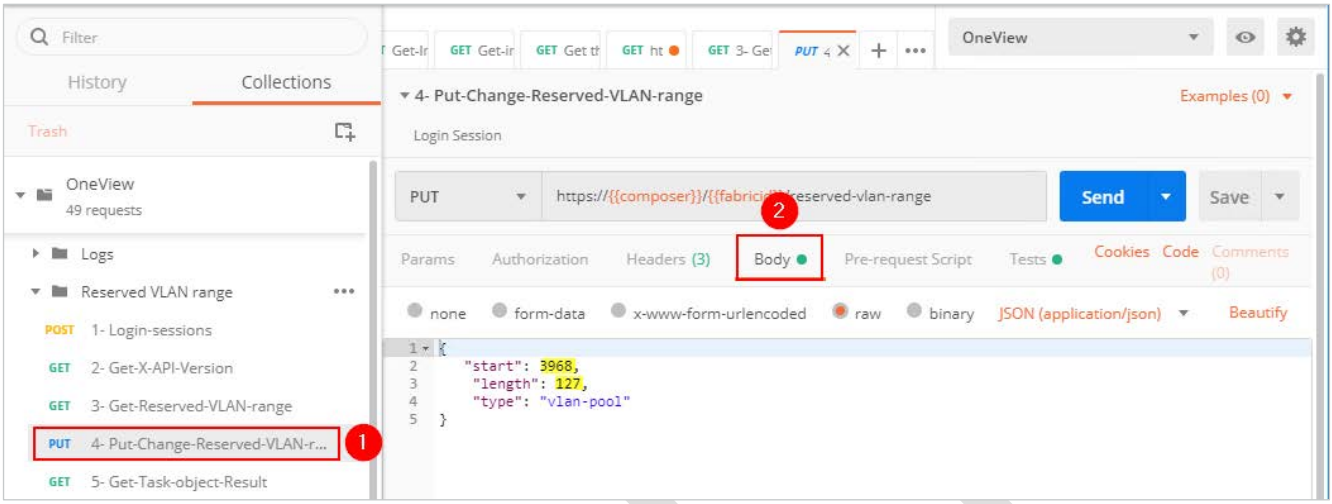
- Next you can show the Reserved VLAN range set in OneView, this information can only be accessible from the REST API, in other words, it cannot be found in the GUI. Expand **Reserved VLAN range**, select **3-Get-Reserved-VLAN-range** then press **Send**



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-HPOVReservedVlanRange -start 4035 -Length 60`

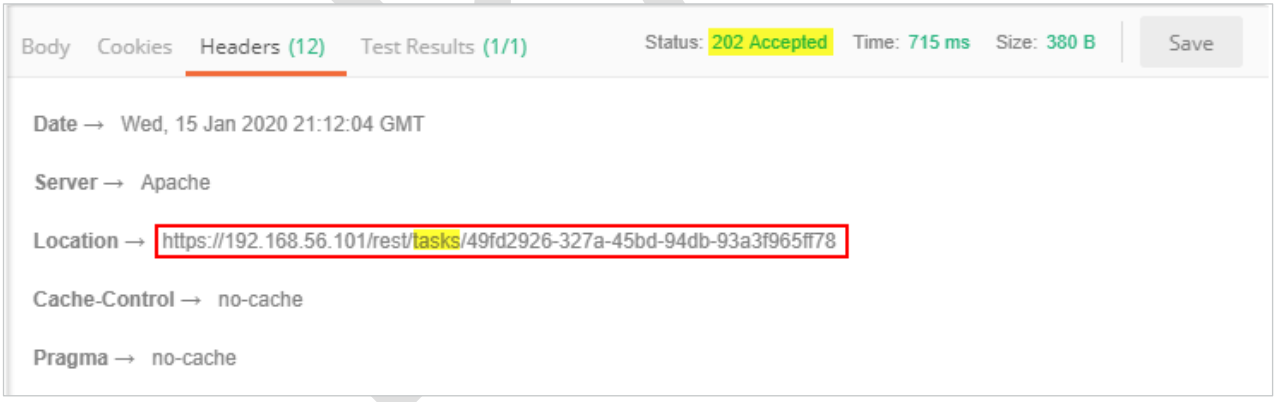
- To demo how to reduce the reserved range, you can run the next REST query in the list: **4- Put-Change-Reserved-VLAN-range** then select **Body**



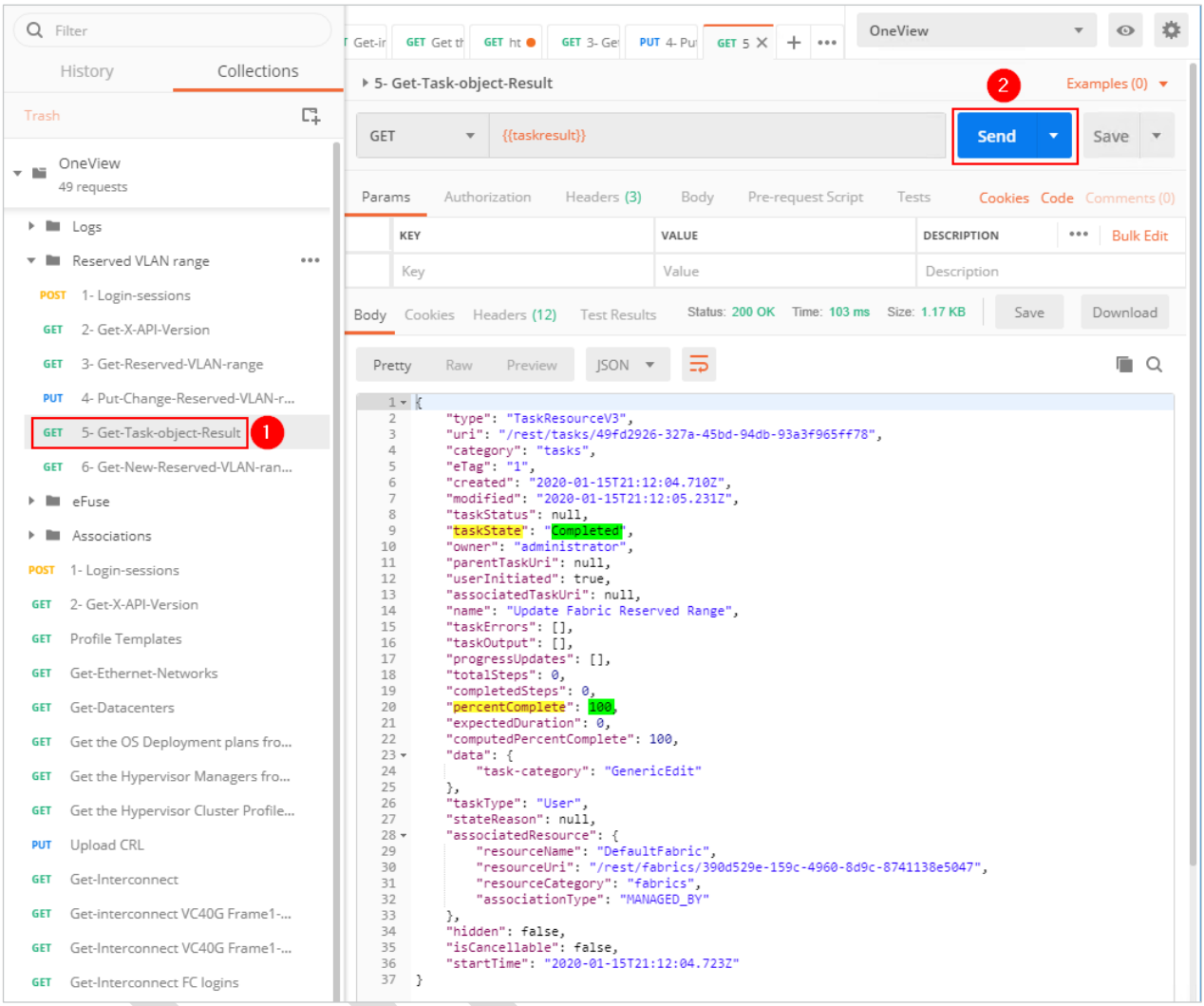
- This example shows how to reduce the reserved VLAN to 127 VLANs and set the reserved range to [3968-4094] releasing VLAN 3967 for other use like in a Cisco ACI environment.

> **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.

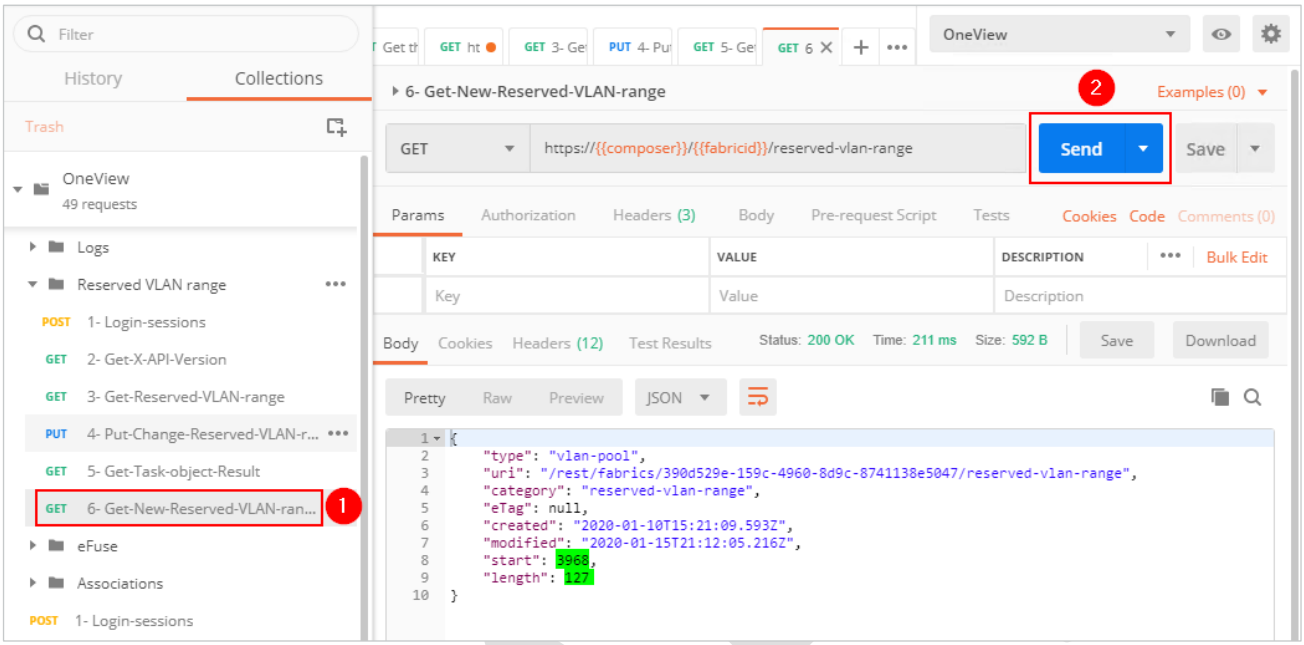


- Select the next REST call in the list **5- Get-Task-object-Result** to query this task ID, press **Send**



- You should see the task completion in the response body

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



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

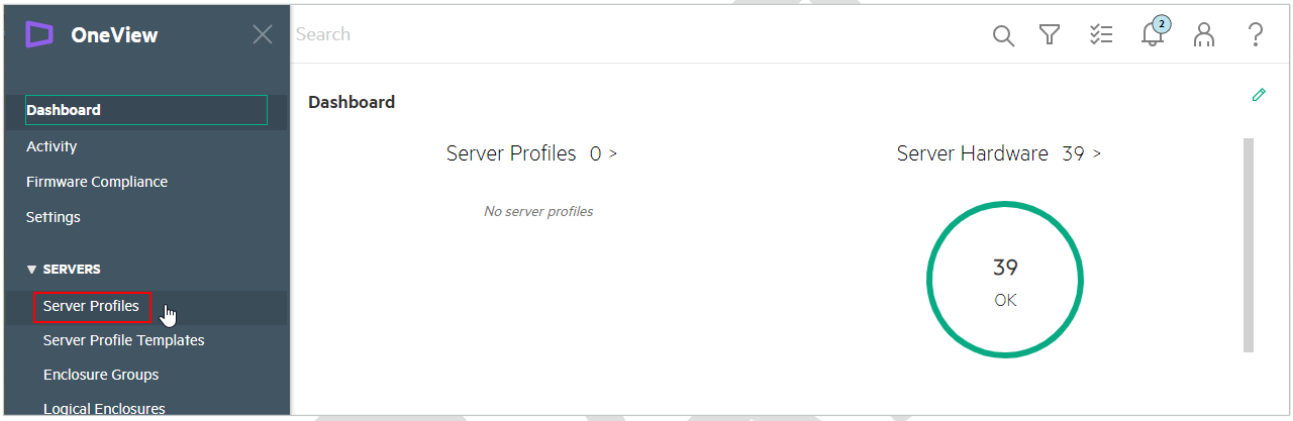
This concludes Postman – scenario 1

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

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

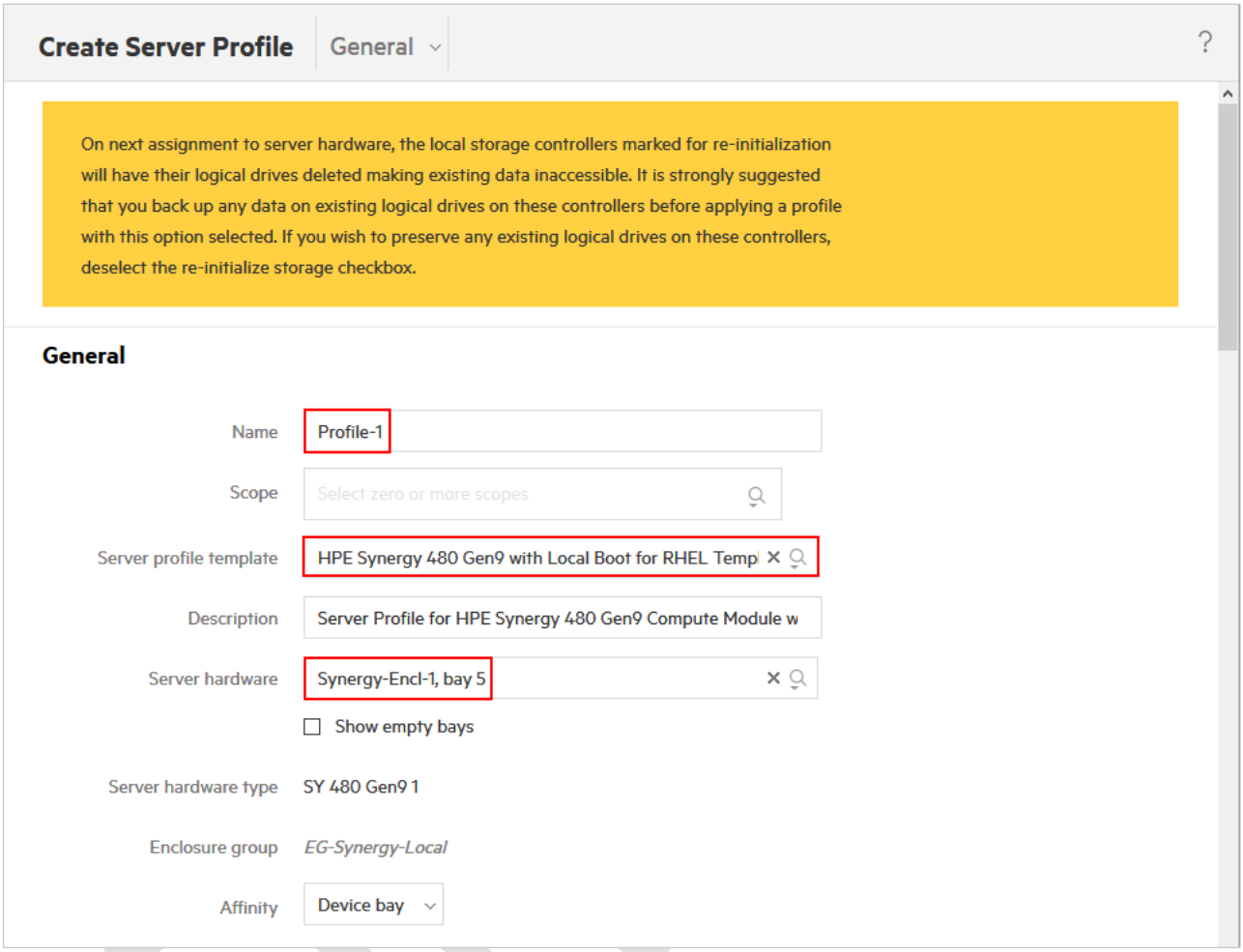
> **Import notice**: This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)

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



- Click on **Create profile**

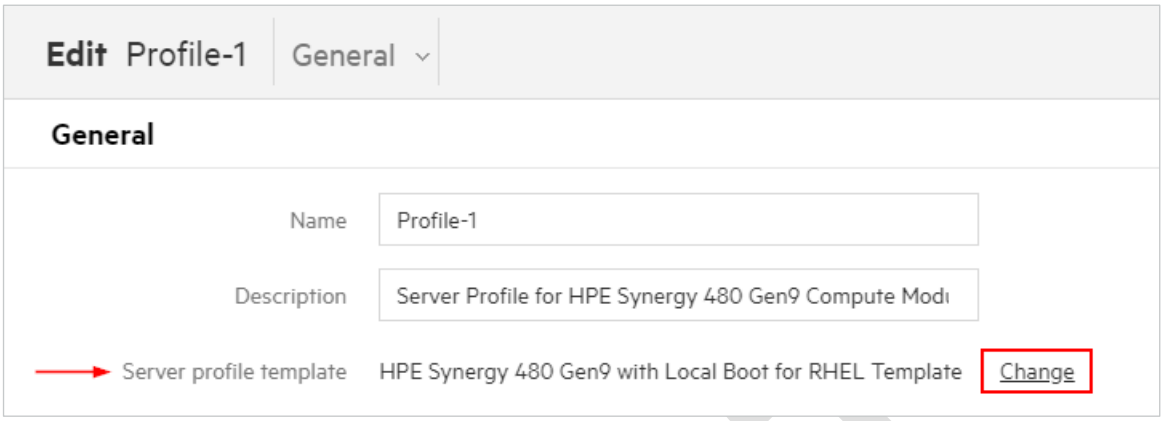
- Name the profile **Profile-1**, select the Server Profile Template **HPE Synergy 480 Gen9 with Local Boot for RHEL Template** and use the server hardware **Synergy-Encl-1, bay 5**:



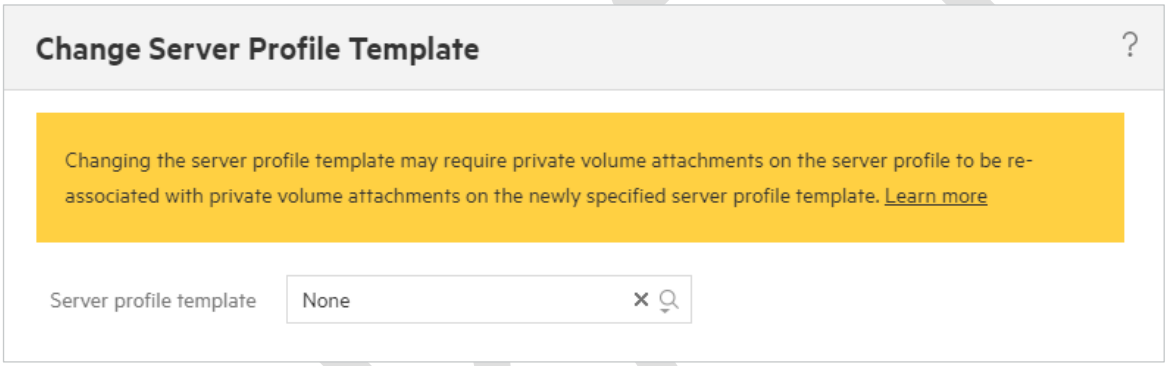
- Then click **Create**

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

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



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



- Open VS Code using the **Synergy demonstrations on Windows** workspace
- Copy/paste the following commands in the console to connect to OneView:

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

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

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

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

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

- A OneView connection confirmation should be displayed:

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> #IP address of OneView

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $IP = "192.168.56.101"

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

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> # OneView Credentials

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $username = "Administrator"

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $password = "password"

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

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> $credentials = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)

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

PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> Connect-HPOVMgmt -appliance $IP -Credential $credentials

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


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


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

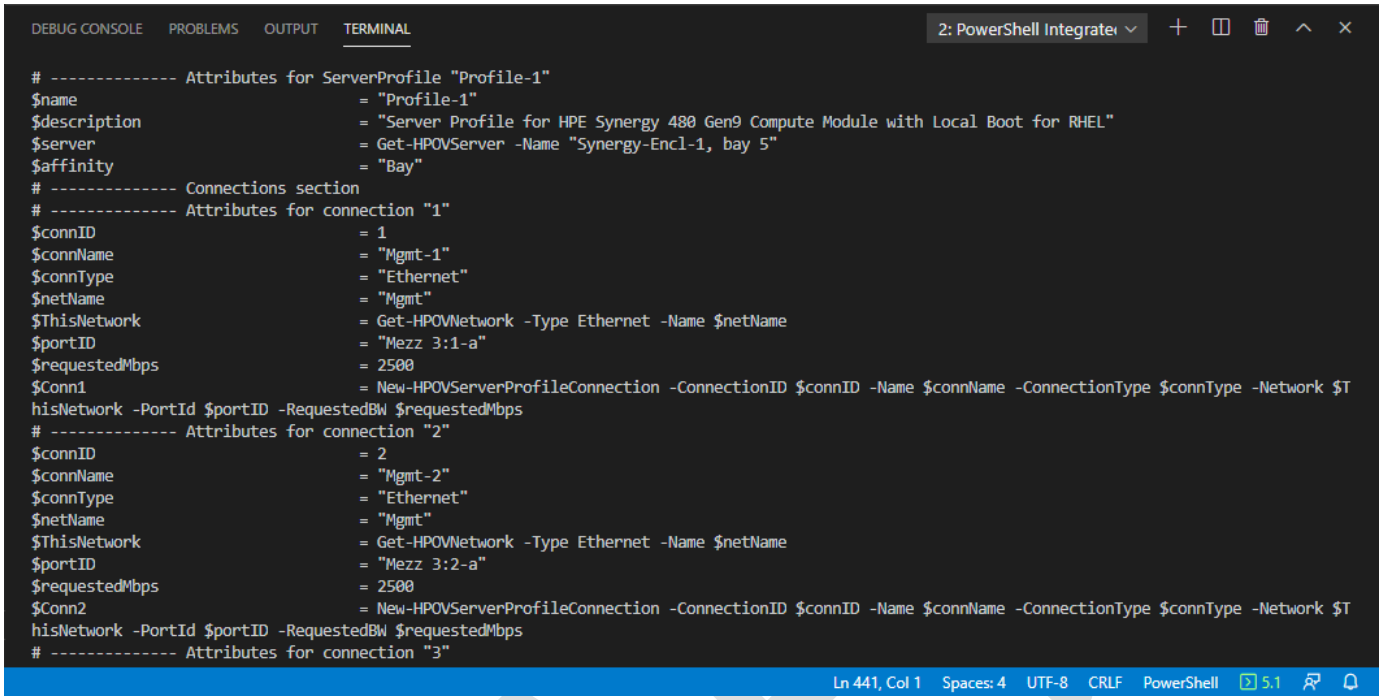
- Next, we can use the `get-hpovserverprofile` cmdlet to get the list of server profiles. To reduce the response to our server profile, we can enter:

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

```
PS C:\VS Code projects\Repositories\Populate-HPE-Synergy> get-hpovserverprofile -name Profile-1


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


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

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

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



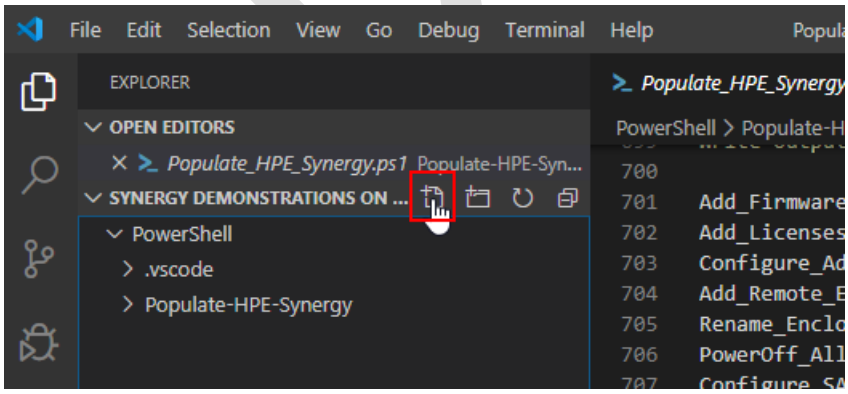
Explain to the customer that this cmdlet can assist administrators or scripters to help generate script code from specific resources. The code generated is a starting point to be used for repeating similar tasks performed by the UI, or to incorporate into scripts or workflows.

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

- Lastly, disconnect the PowerShell client to the appliance:

```
Disconnect-HPOVMgmt
```

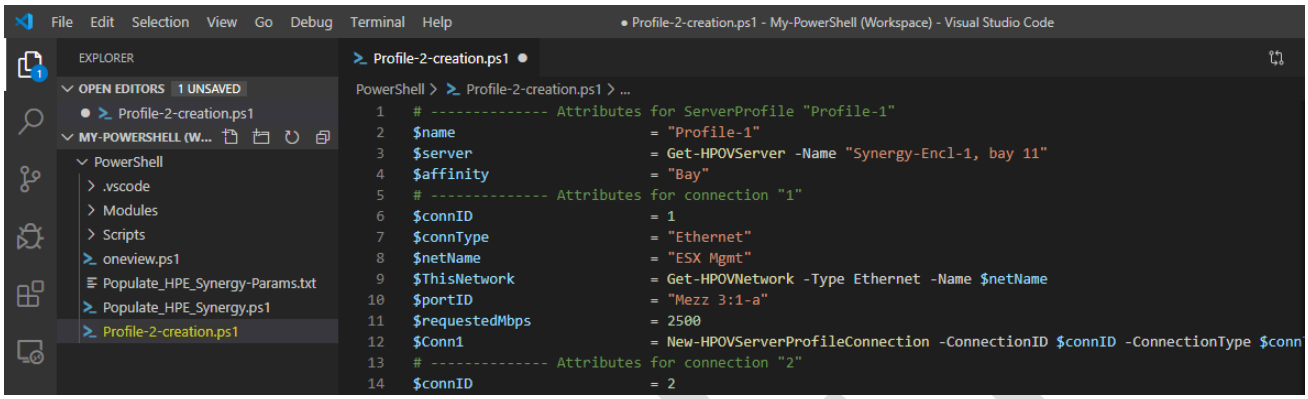
- Next, we can demonstrate how we can easily create a script using this generated code to create a new Server Profile in OneView. This will require just a few modifications. Click on **New File** to create a new script:



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

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

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



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

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

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

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

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

- Then change the following values:
    - `$name` with **Profile-2**
    - `$server` with **Synergy-Encl-2, bay 5**
    - The requested bandwidth of connection **1** and **2** to **7500** Mbps

Populate_HPE_Synergy.ps1 ● Profile-2-creation.ps1 ●

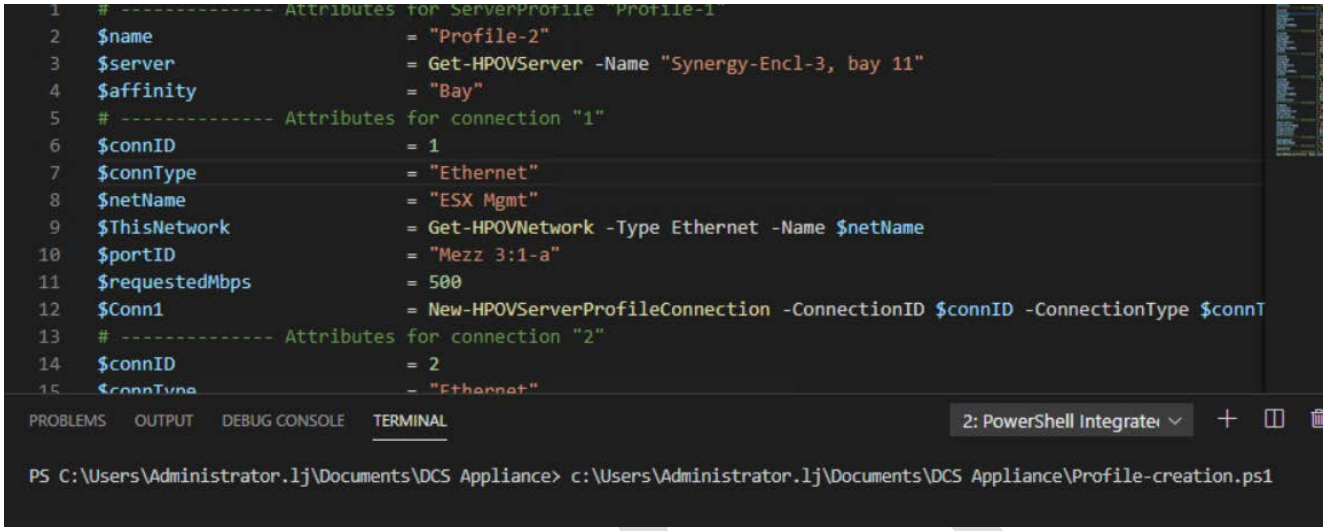
PowerShell > Profile-2-creation.ps1 > ...

```
# 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

# -------------- Attributes for ServerProfile "Profile-2"
$name                           = "Profile-2"
$description                    = "Server Profile for HPE Synergy 480 Gen9 Compute Module with Local Boot for RHEL"
$server                         = Get-HPOVServer -Name "Synergy-Encl-2, bay 5"
$affinity                       = "Bay"
# -------------- Connections section
# -------------- Attributes for connection "1"
$connID                         = 1
$connName                       = "Mgmt-1"
$connType                       = "Ethernet"
$netName                        = "Mgmt"
$ThisNetwork                    = Get-HPOVNetwork -Type Ethernet -Name $netName
$portID                         = "Mezz 3:1-a"
$requestedMbps                  = 7500
$Conn1                          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy
# -------------- Attributes for connection "2"
$connID                         = 2
$connName                       = "Mgmt-2"
$connType                       = "Ethernet"
$netName                        = "Mgmt"
$ThisNetwork                    = Get-HPOVNetwork -Type Ethernet -Name $netName
$portID                         = "Mezz 3:2-a"
$requestedMbps                  = 7500
$Conn2                          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy
# -------------- Attributes for connection "3"
$connID                         = 3
$connName                       = "Prod-NetworkSet-1"
$connType                       = "Ethernet"
$netName                        = "Prod"
$ThisNetwork                    = Get-HPOVNetworkSet -Name $netName
$portID                         = "Mezz 3:1-c"
$requestedMbps                  = 2500
$Conn3                          = New-HPOVServerProfileConnection -ConnectionID $connID -Name $connName -ConnectionTy
# -------------- Attributes for connection "4"
```

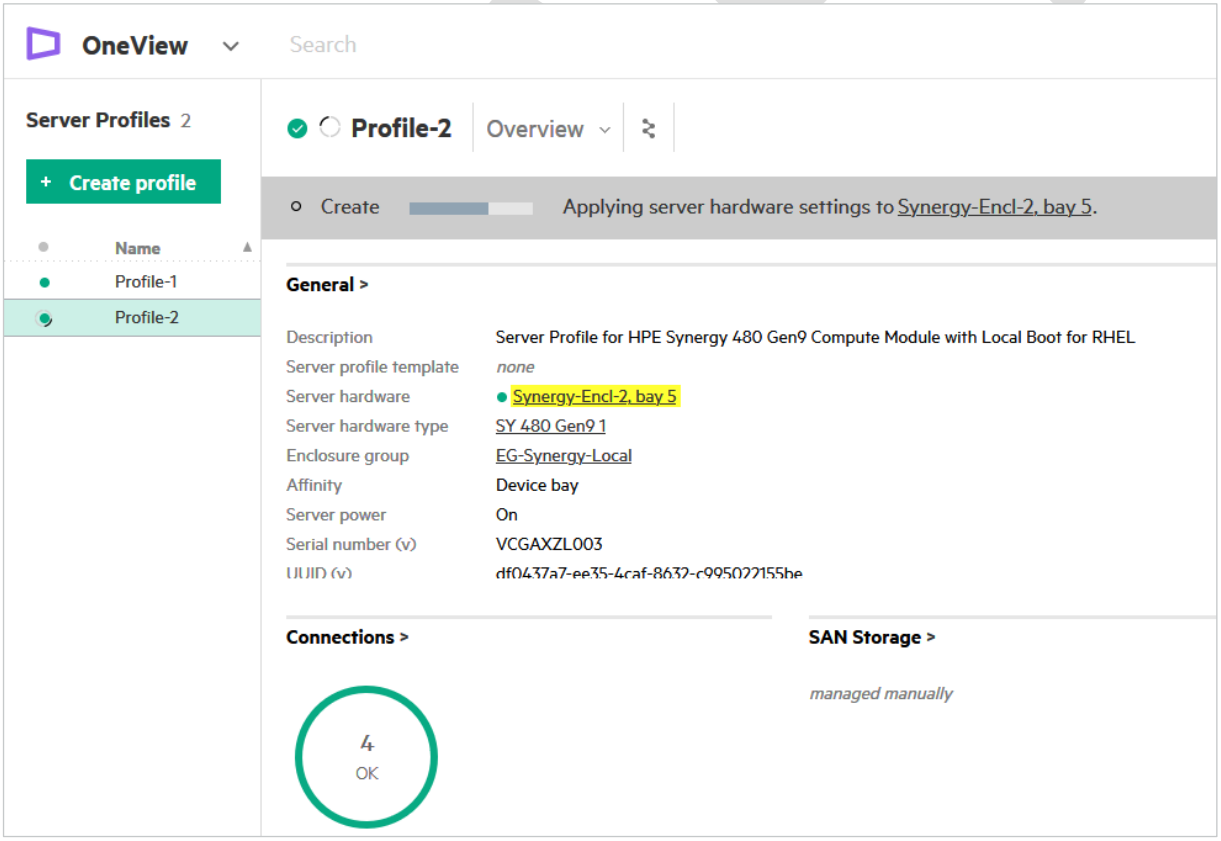
- Then add the following line at the end of the script to disconnect properly the client to the appliance:

```
Disconnect-HPOVMgmt
```

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

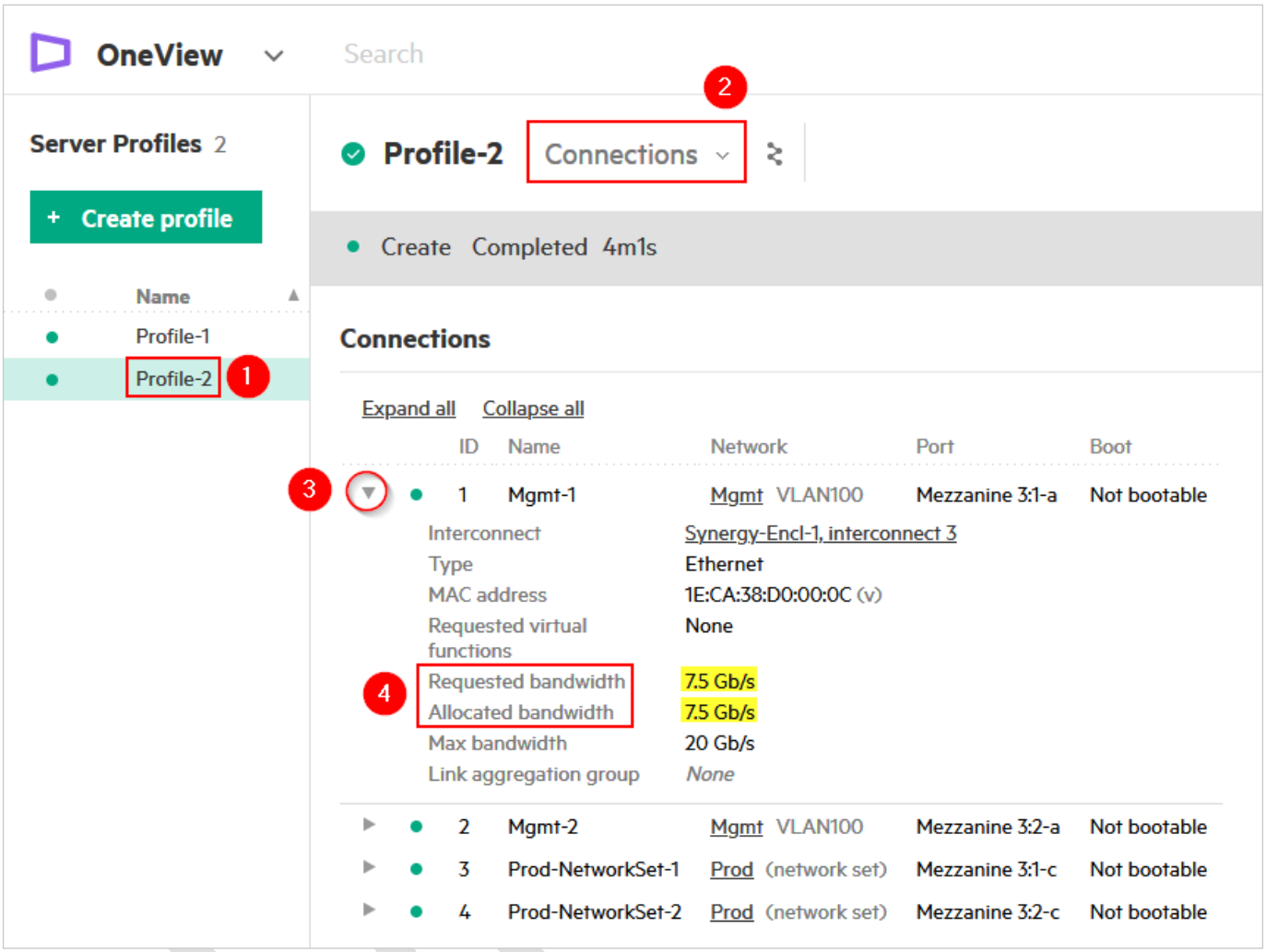


- You can then jump to the OneView GUI to show the creation of Profile-2



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

- To show the new profile parameters that have been changed, click on the **Profile-2**, select **Connections** in the profile menu, expand connection **1** then show the 7500Mbps (*7.5Gps)* requested bandwidth that was defined in our code:



This concludes PowerShell – Scenario 1

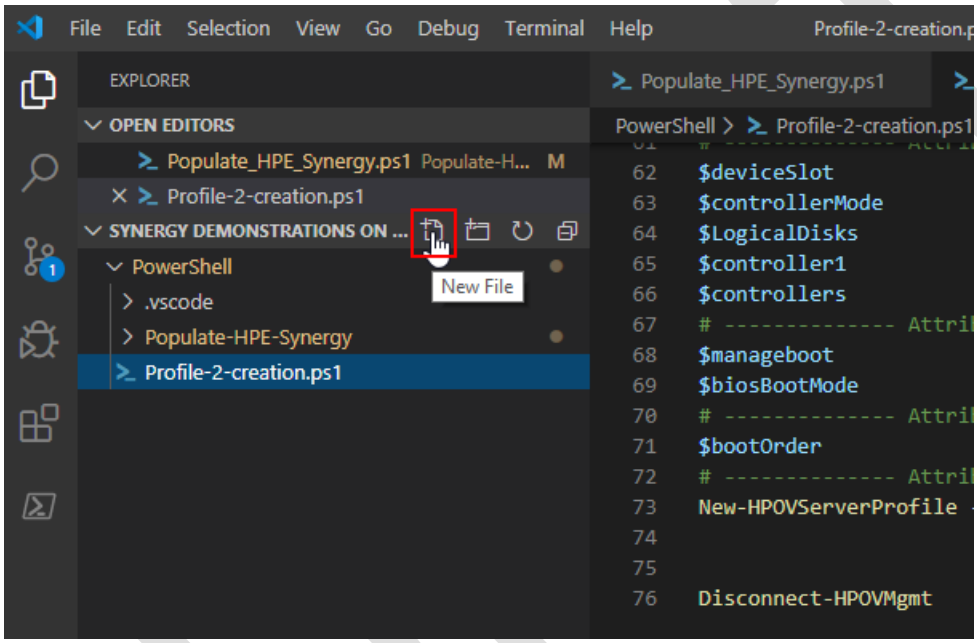
PowerShell – Scenario 2 - Creating a report

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

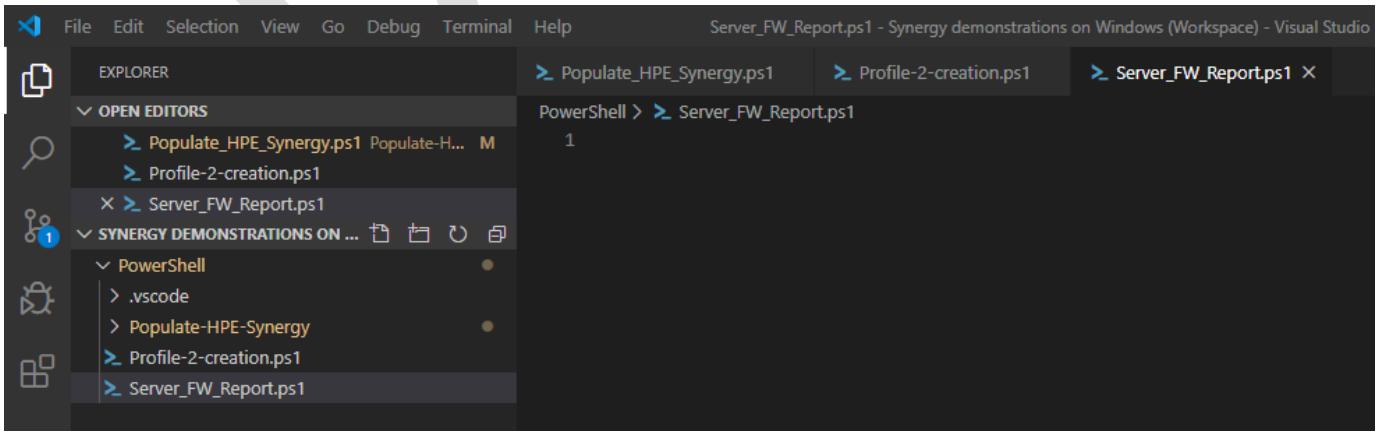
Server Name	Rom Version	Component Name	Component Firmware Version
Server1	P89 v2.42 (04/25/2017)	HPE Smart Storage Battery 1	Firmware 1.1
Server1	P89 v2.42 (04/25/2017)	Intelligent Platform Abstraction	Data 25.05
Server1	P89 v2.42 (04/25/2017)	Smart Array P440ar Controller	2.40

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

- Create a new file in VS Code



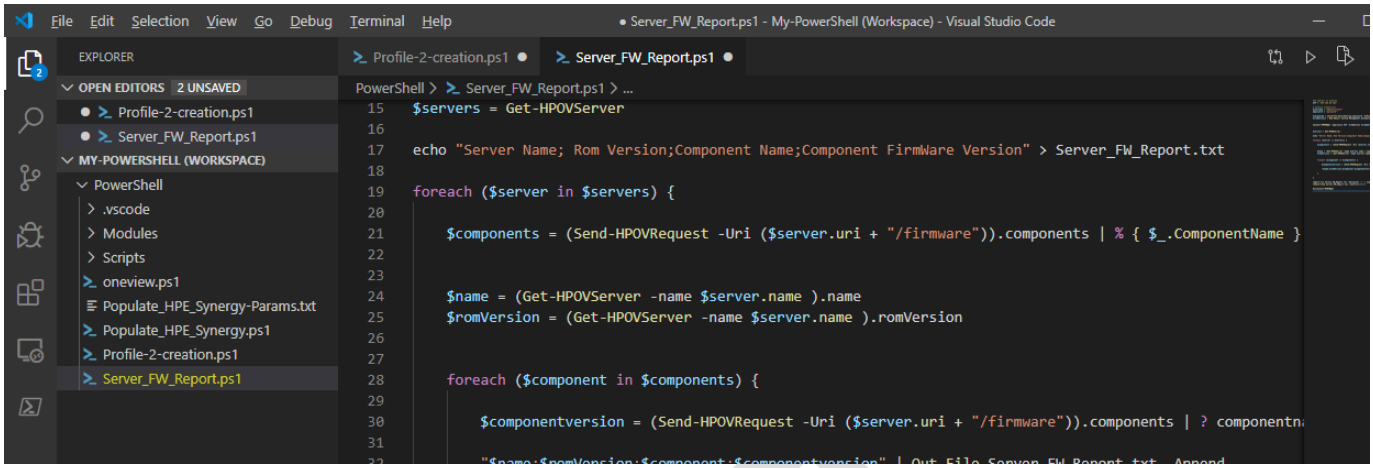
- Name it **Server_FW_Report.ps1**



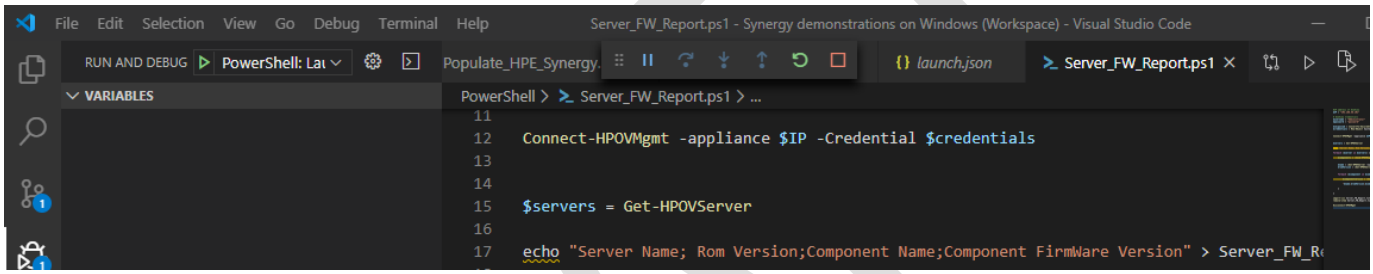
- Copy/paste the following script content in the new VS Code file:

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

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

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


Connect-HPOVMgmt -appliance $IP -Credential $credentials


$servers = Get-HPOVServer

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

foreach ($server in $servers) {

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


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


    foreach ($component in $components) {

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

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

    }

}

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

Disconnect-HPOVMgmt
```

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

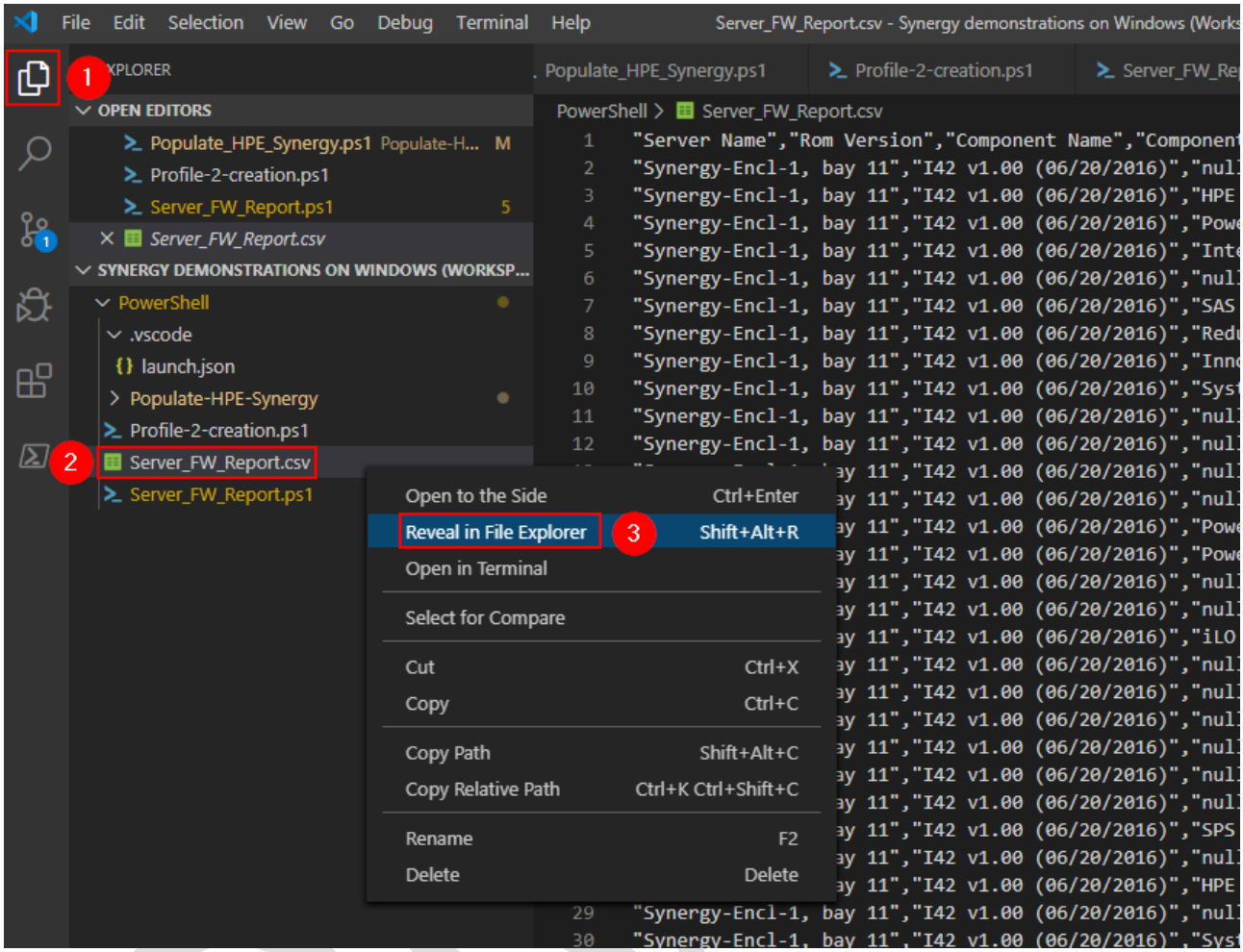


- Then we can run the script by pressing **F5**

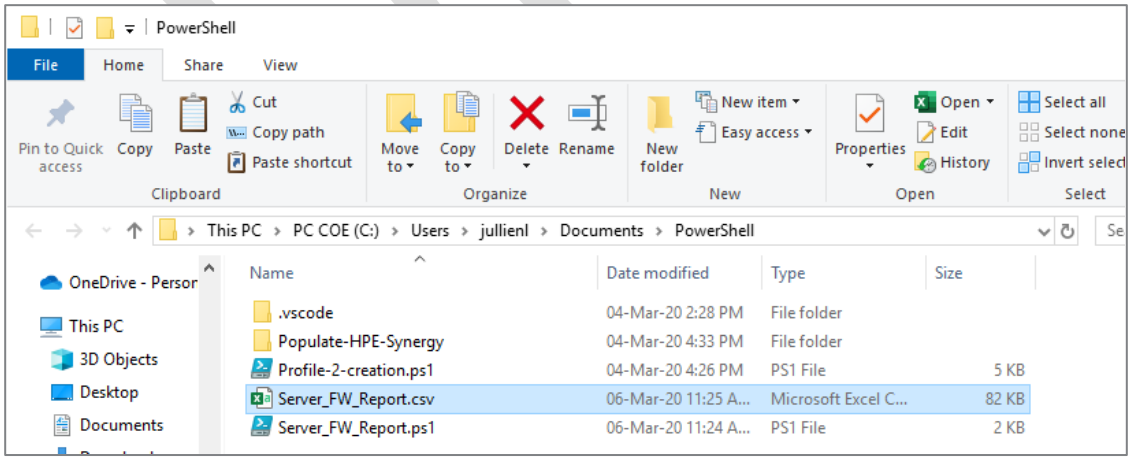


The script generates a CSV file in your working folder

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



This will open the Windows explorer



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

A1 | Server Name

	A	B	C	
34	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null	2.11.5.13-3.sles15sp0 6.0.0.0-4 15.6.1.3 3.40-3
35	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	ME SPI Descriptor	1.1 0
36	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null	2.11.5.13-3.sles15sp0 6.0.0.0-4 15.6.1.3 3.40-3
37	Synergy-Encl-1, bay 11	I42 v1.00 (06/20/2016)	null	2.11.5.13-3.sles15sp0 6.0.0.0-4 15.6.1.3 3.40-3
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	I39 v1.30 08/26/2014
40	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	System Programmable Logic Device	Version 0x2F
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	1.0.9
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	3.0.6.267.1
47	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	HP ProLiant System ROM	I39 v1.30 08/26/2014
48	Synergy-Encl-1, bay 3	I39 v1.30 08/26/2014	iLO	2.30 Jul 23 2015
49	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	System Programmable Logic Device	Version 0x2F
50	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Power Management Controller Firmware	1.0.9
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	I39 v1.30 08/26/2014
56	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	iLO	2.30 Jul 23 2015
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	I39 v1.30 08/26/2014
59	Synergy-Encl-1, bay 4	I39 v1.30 08/26/2014	Server Platform Services (SPS) Firmware	3.0.6.267.1
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	2.30 Jul 23 2015
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	I39 v1.30 08/26/2014
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	I37 v1.30 08/26/2014
66	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	System Programmable Logic Device	Version 0x2F
67	Synergy-Encl-1, bay 5	I37 v1.30 08/26/2014	Server Platform Services (SPS) Firmware	3.0.6.267.1
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	1.0.9

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*.

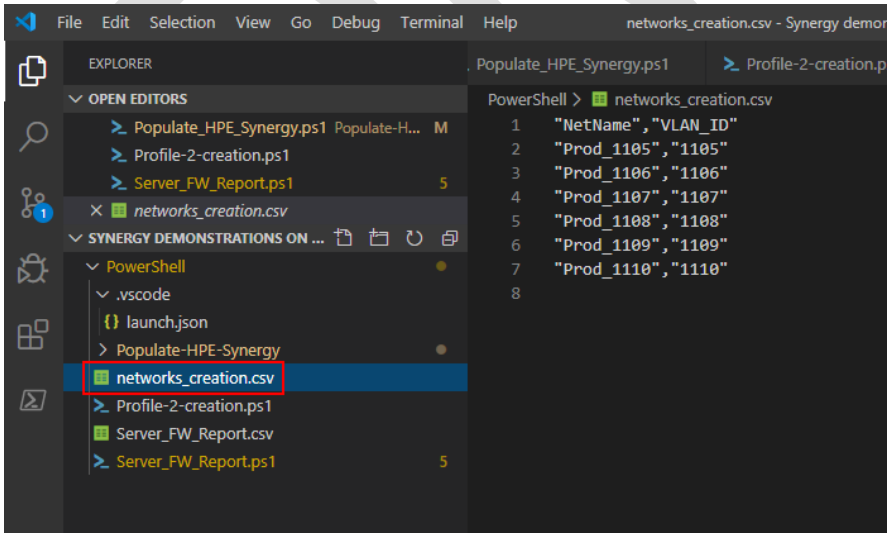
> **Import notice**: This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)

- 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:

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

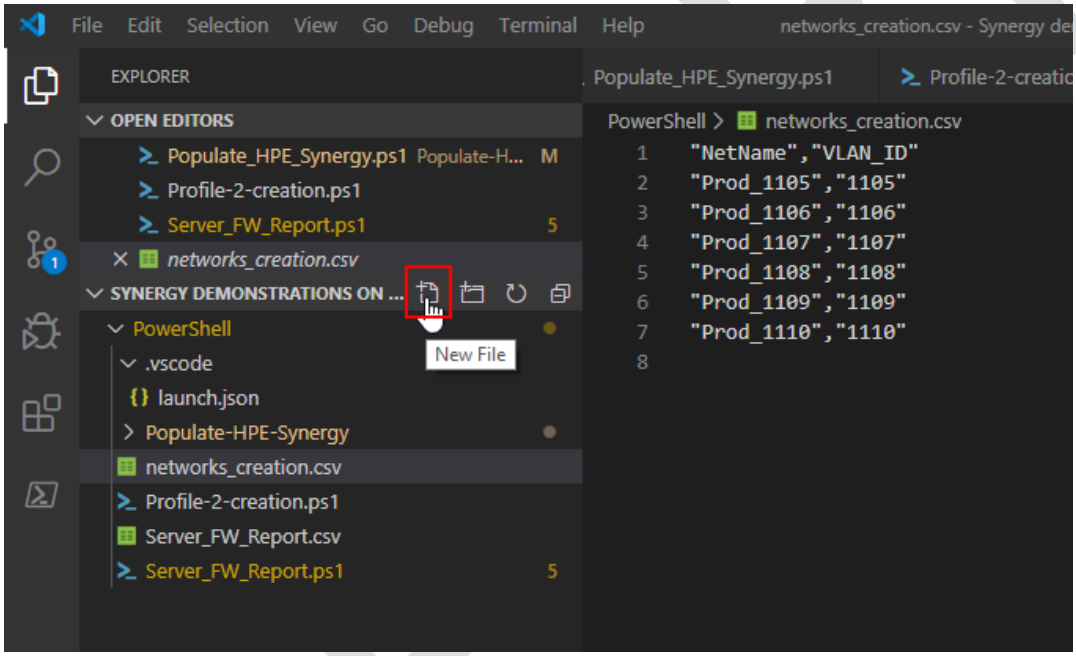


In Excel, it is built as follows:

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

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

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



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

```
$csvfile = "networks_creation.csv"

$IP = "192.168.56.101"
$username = "Administrator"
$password = "password"

$LIG_UplinkSet = "Prod"
$networksetname = "Prod"
```

```
$LIGname = "LIG-FlexFabric"

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

$data = (Import-Csv $csvfile)

# Creating Networks and adding them to the LIG uplink Set

$LIG = Get-HPOVLogicalInterconnectgroup -Name $LIGname

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

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

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

}

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

# Updating LI from LIG

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

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

    if ($Interconnectstate -notcontains "Configured") {

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

}

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

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

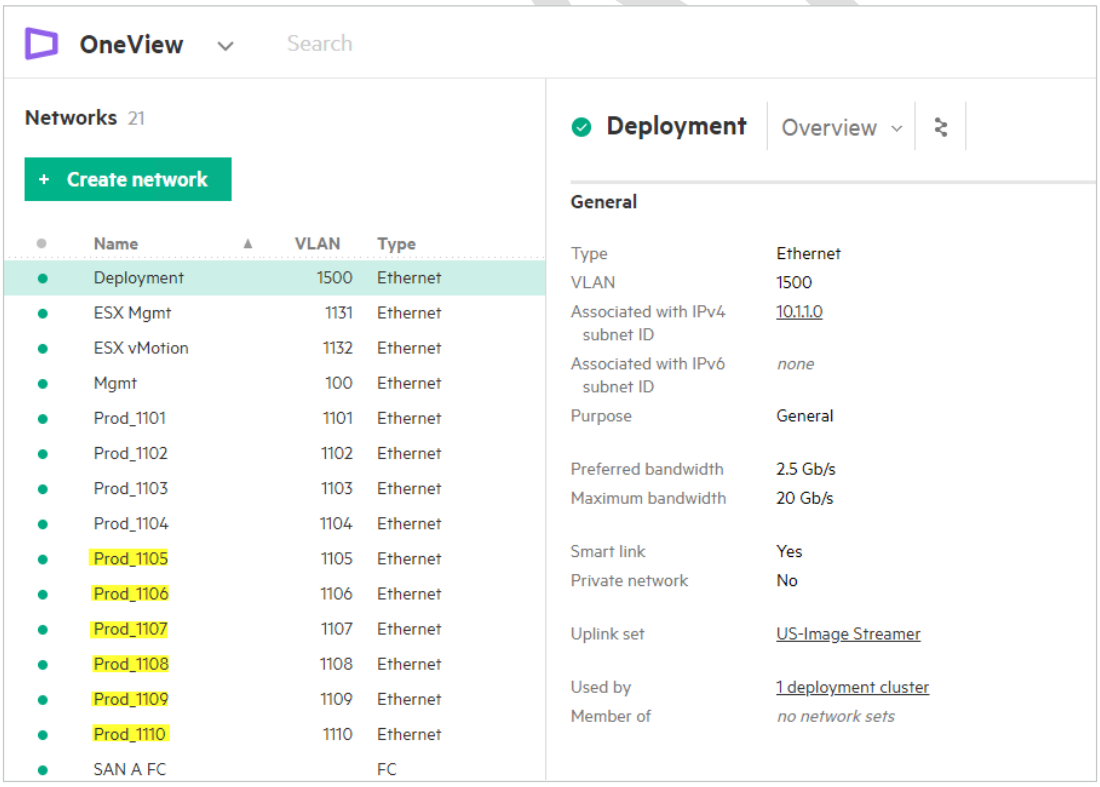
```
try {
    Get-HPOVLogicalInterconnect -Name $LI.name | Update-HPOVLogicalInterconnect -confirm:$false -
ErrorAction Stop | Wait-HPOVTaskComplete | Out-Null
}
catch {
    Write-Output $_ #.Exception
}

# Adding Network to Network Set

ForEach ($VLAN In $data) {

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

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

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

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

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

}

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

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

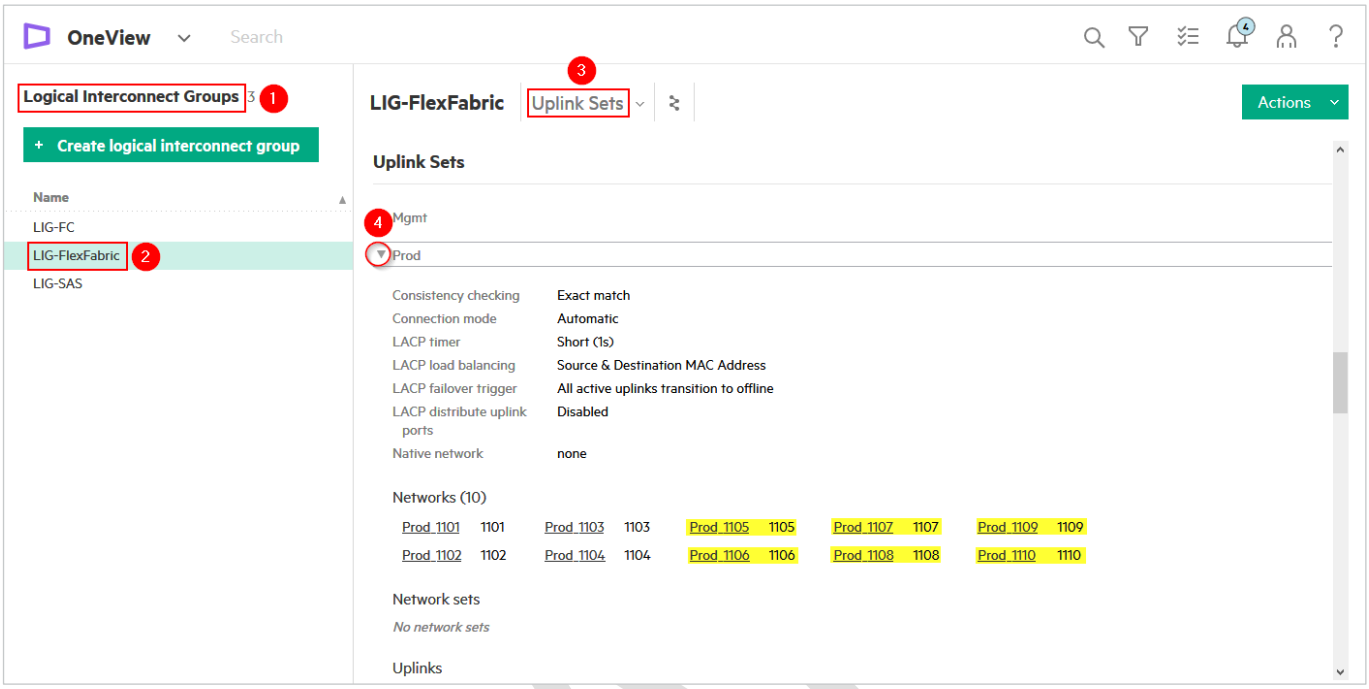
- The first step of the script is the creation of the six networks imported from the CSV file in OneView:

```
DEBUG CONSOLE   PROBLEMS 1   OUTPUT   TERMINAL

Adding Network: Prod_1105 to Uplink Set: Prod

Update from group LE-Synergy-Local-LIG-FlexFabric
   Uplink set changed.ct module state
   [oooooo
   [ooooooooooooooo
Creating Network: Prod_1108
Adding Network: Prod_1108 to Uplink Set: Prod

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

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


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

Please wait...
```

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



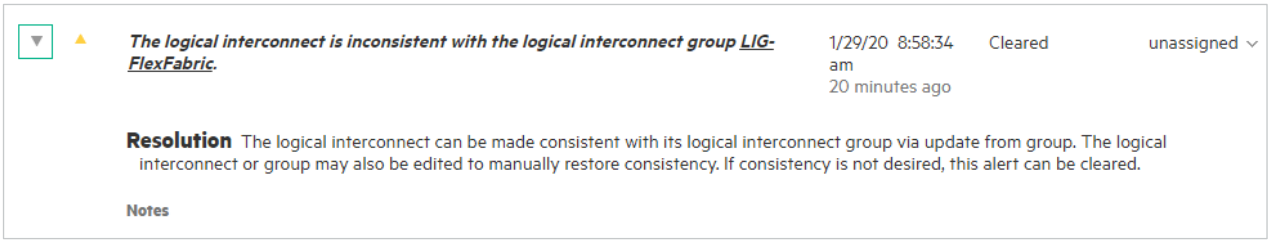
- In the Logical Interconnect Group, show that *LIG-FlexFabric* is now defined with 6 new networks in the *Prod* uplink set:



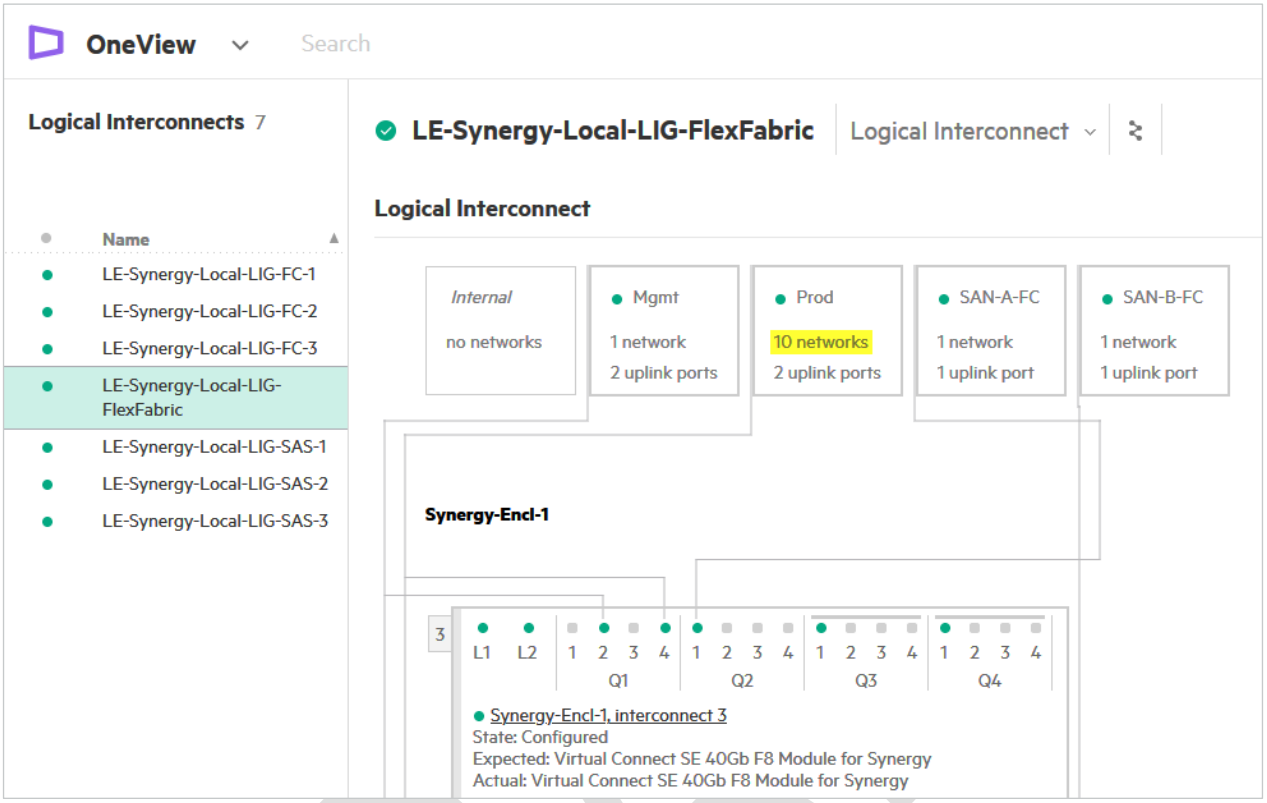
- Back to VS Code, the next step is the update of our Logical Interconnect from its Logical Interconnect Group (LIG-*FlexFabric*):

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

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

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


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

Please wait...
```

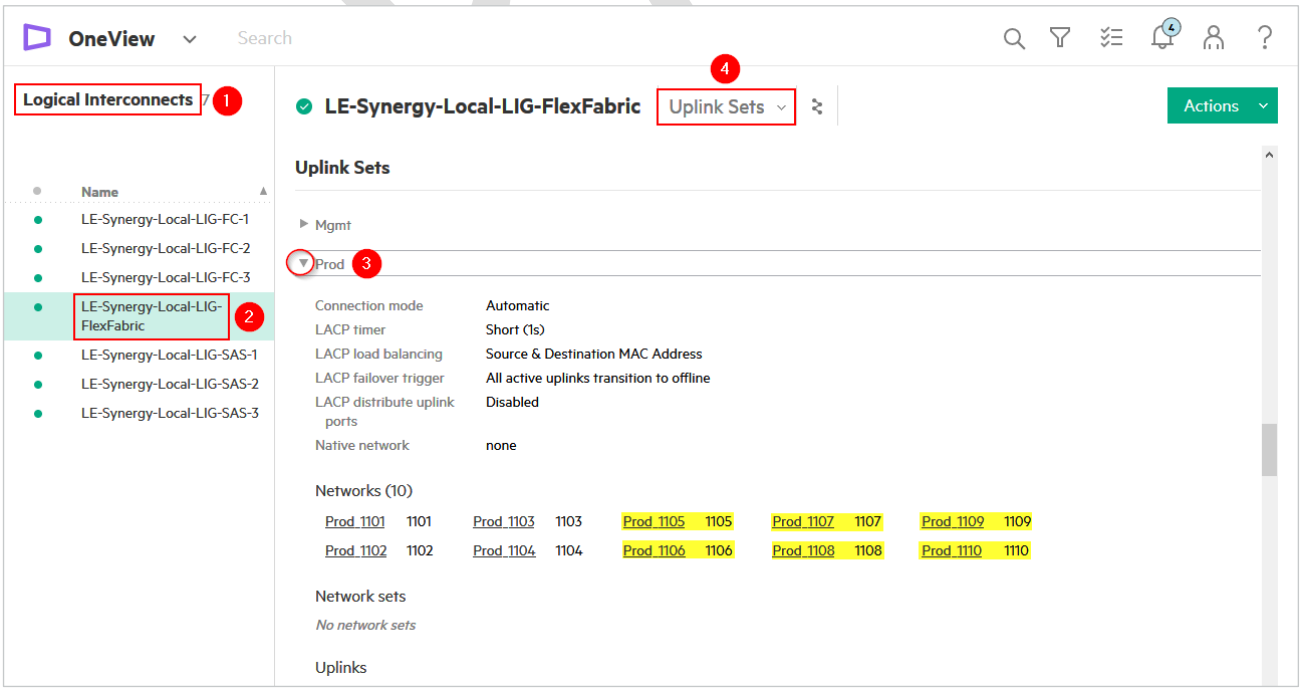
- This step takes some time (4-5mn) so take the opportunity to explain the concept of LI/LIG, show that the LI is inconsistent with the logical Interconnect group (shown in *Activity*):



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



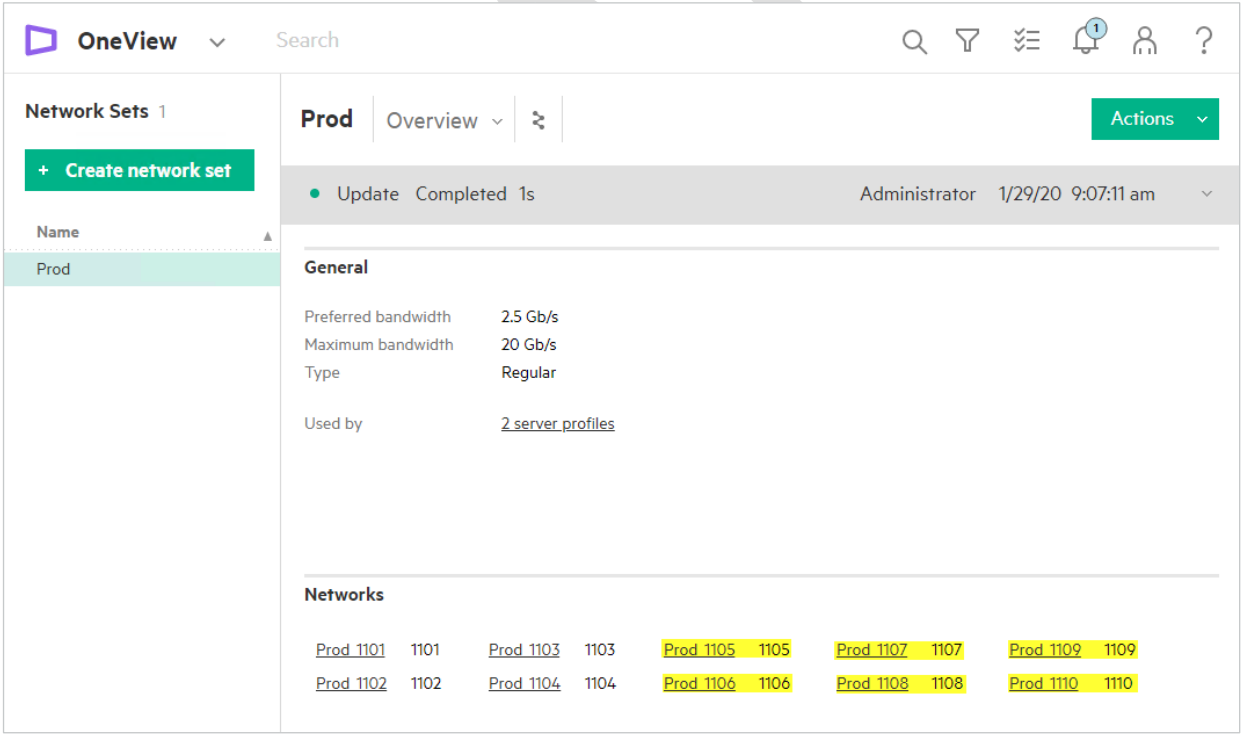
- You can then show the new networks appearing in the *US-Prod* uplink Set:



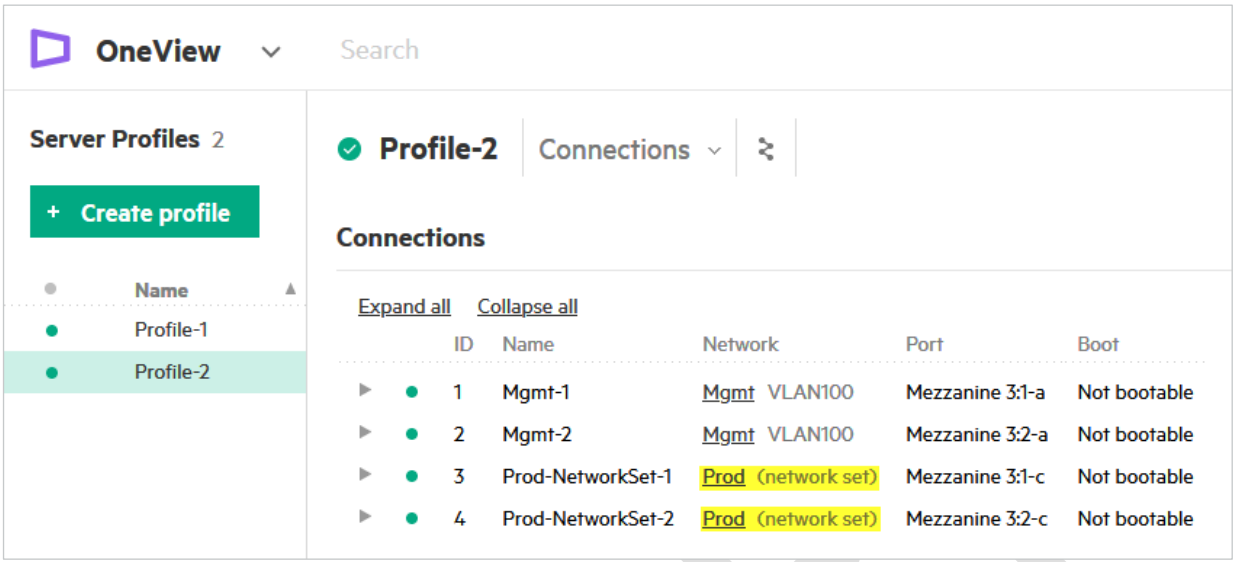
- In the next step, the script is adding the 6 networks to the Network set *Prod*

```
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



- 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:



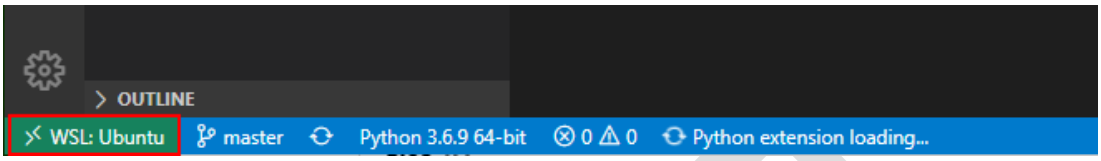
This concludes PowerShell – Scenario 3

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

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

Prerequisites:

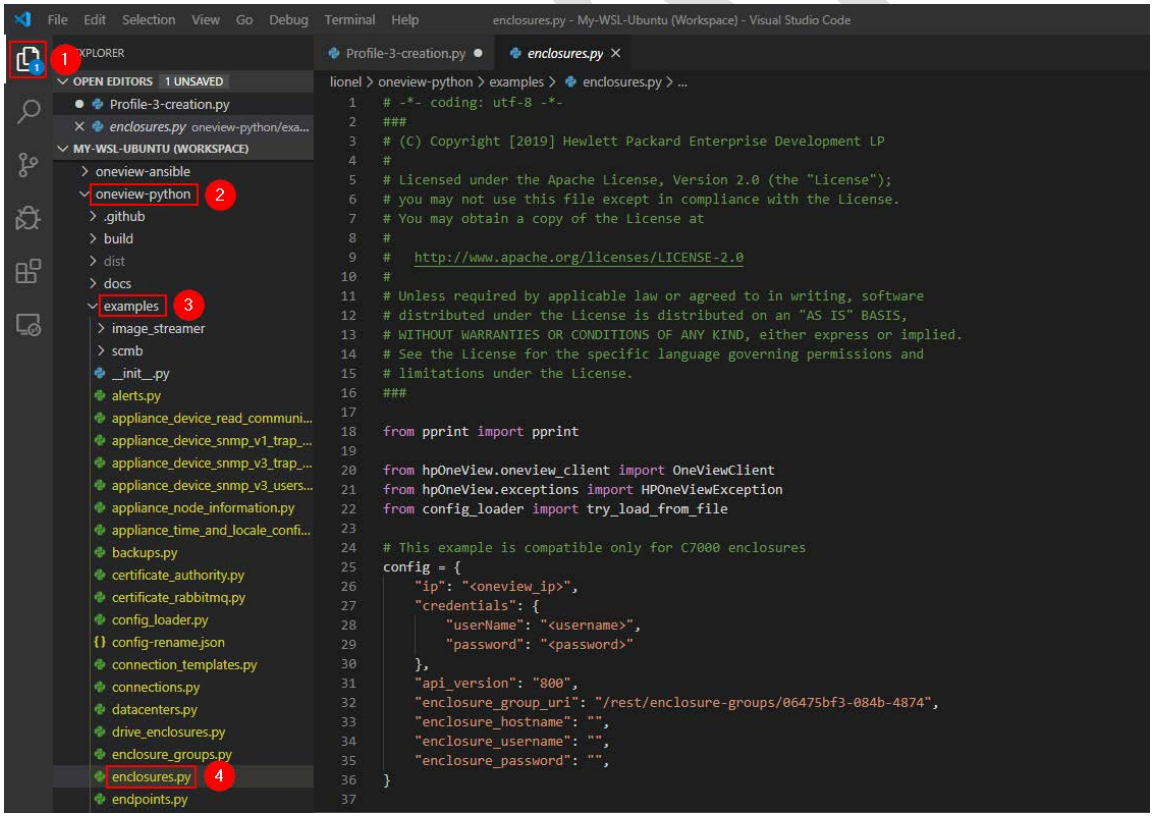
- Open VS Code using **Synergy demonstrations on WSL** workspace, ensure that the Remote WSL extension gets connected by checking the presence of *WSL: Ubuntu* in the status bar in the lower left corner:



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

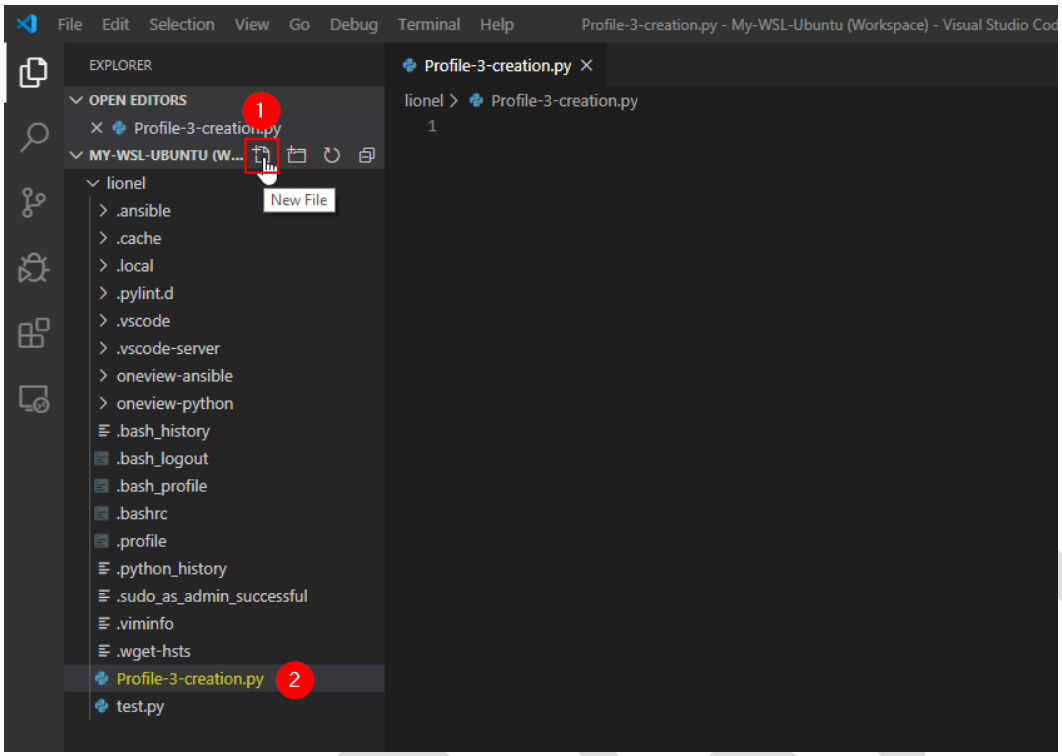
Before creating a server profile, you can show the list of examples provided by the *OneView-Python* library.

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



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

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



- Then add the following lines to your code:

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

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

oneview_client = OneViewClient(config)

server_hardwares = oneview_client.server_hardware

server_profile_templates = oneview_client.server_profile_templates

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

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

profile = myspt.get_new_profile()

profile['serverHardwareUri'] = server.data['uri']
```

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

oneview_client.server_profiles.create(profile)

server_profiles = oneview_client.server_profiles

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

A few explanations:

- Import the HPOneView module with:

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

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

  ```
  from pprint import pprint
  ```

- Provide the OneView information and credentials:

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

- Make the connection with the appliance:

  ```
  oneview_client = OneViewClient(config)
  ```

- Get the server hardware information for later use:

  ```
  server_hardwares = oneview_client.server_hardware
  ```

- Get the server profile template information for later use:

  ```
  server_profile_templates = oneview_client.server_profile_templates
  ```

- Pull the information of our Server Profile Template from *server_profile_templates*:

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

- Pull the information of Server 'Synergy-Encl-3, bay 7' from *server_profile_templates*:

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

- Generate a new profile from our Server Profile Template:

  ```
  profile = myspt.get_new_profile()
  ```

- Modify the 'serverHardwareUri' property of the generated profile object with the value of the selected server hardware uri:

  ```
  profile['serverHardwareUri'] = server.data['uri']
  ```

- Set the server profile name:

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

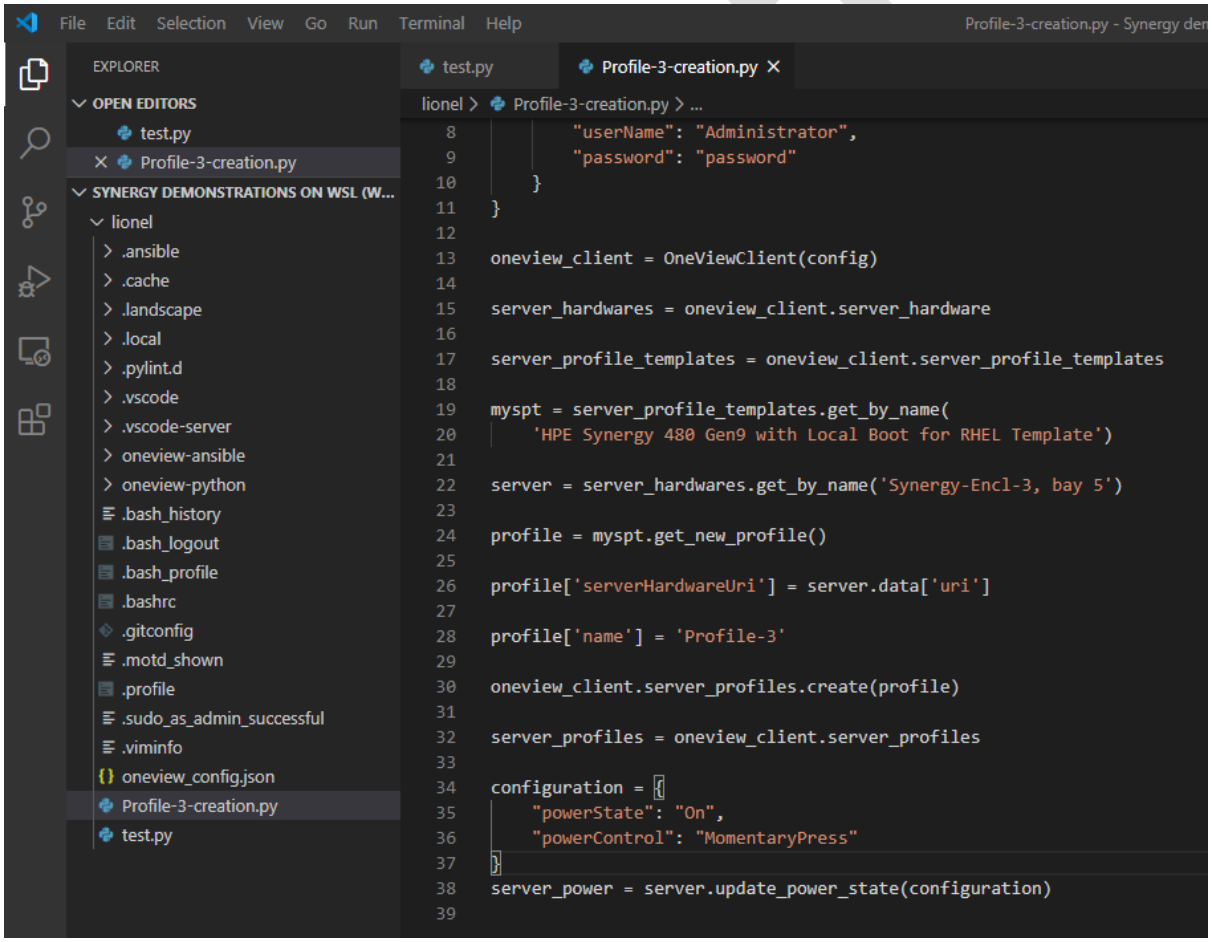
- Create a new profile from our Server Profile Template:

```
oneview_client.server_profiles.create(profile)
```

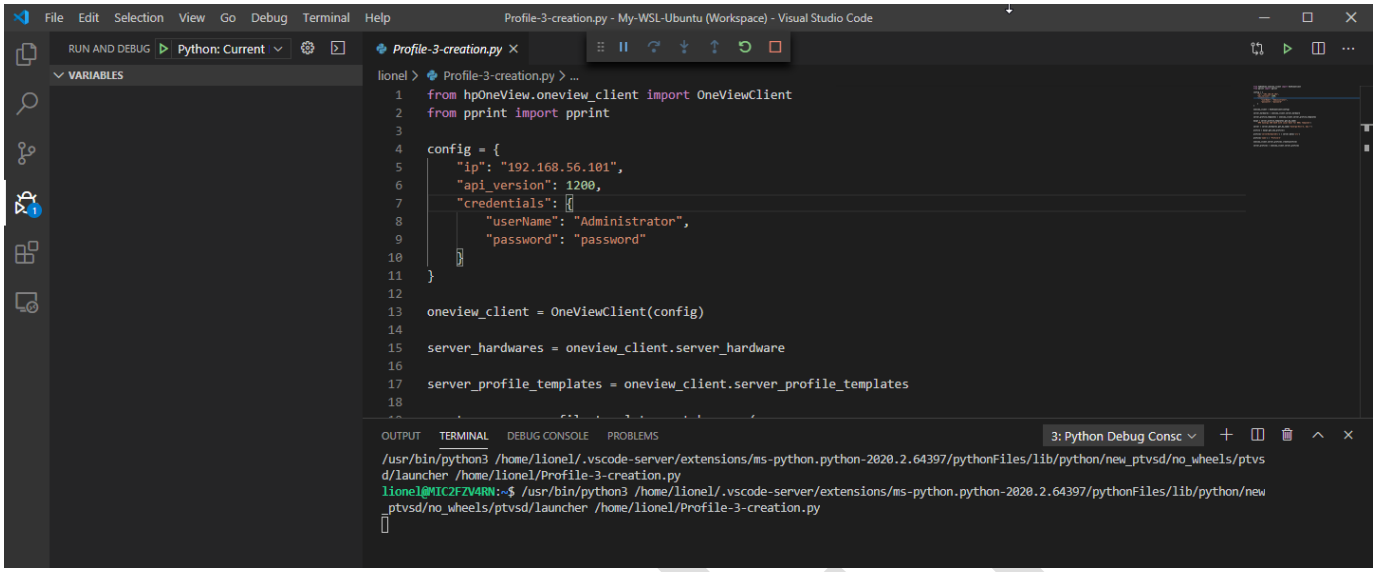
- Turn the server on:

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

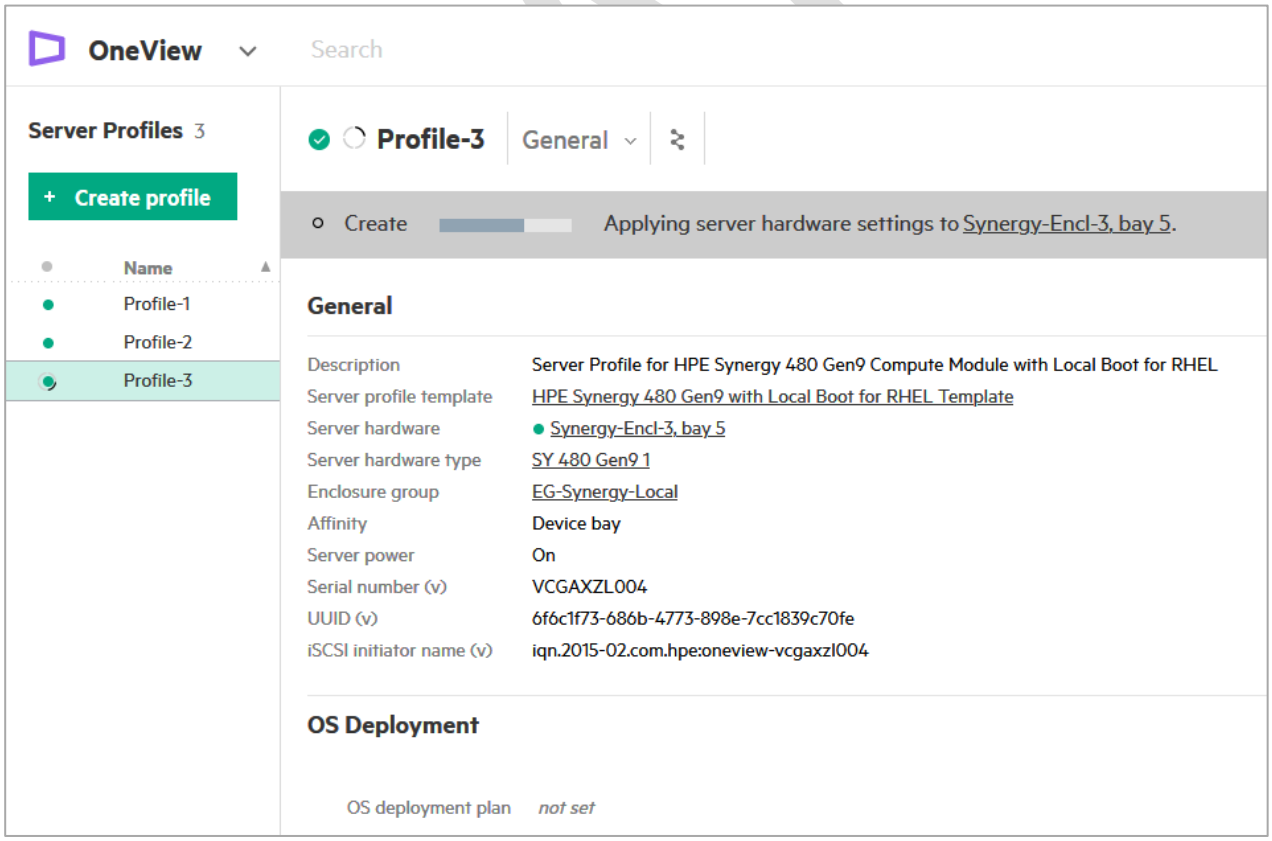
- Save now the script, press **CTRL** + **S**.



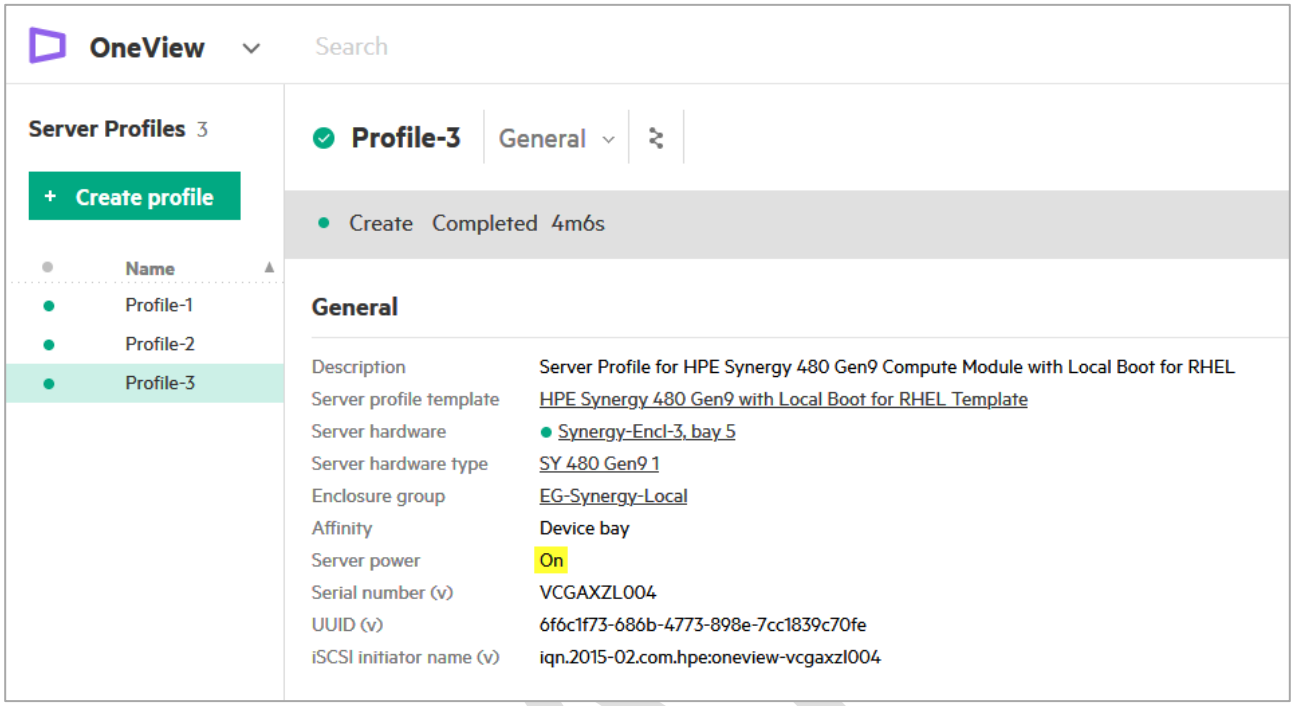
- Then to run the script, press **F5**.



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



- Once created, the server is turned on:



This concludes Python – Scenario 1

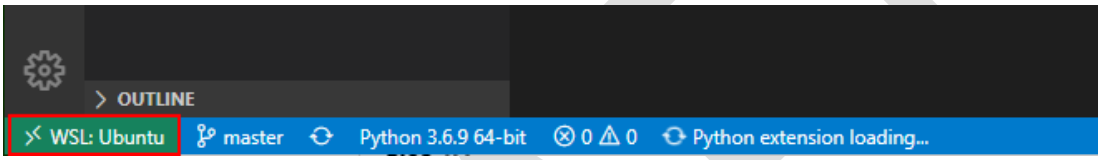
Python – Scenario 2 – Creating a report

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

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

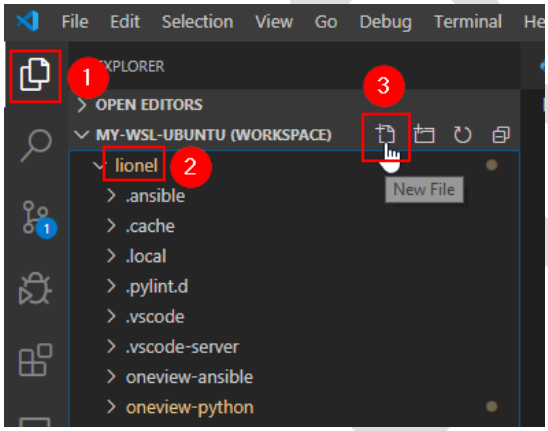
Prerequisites:

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



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

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

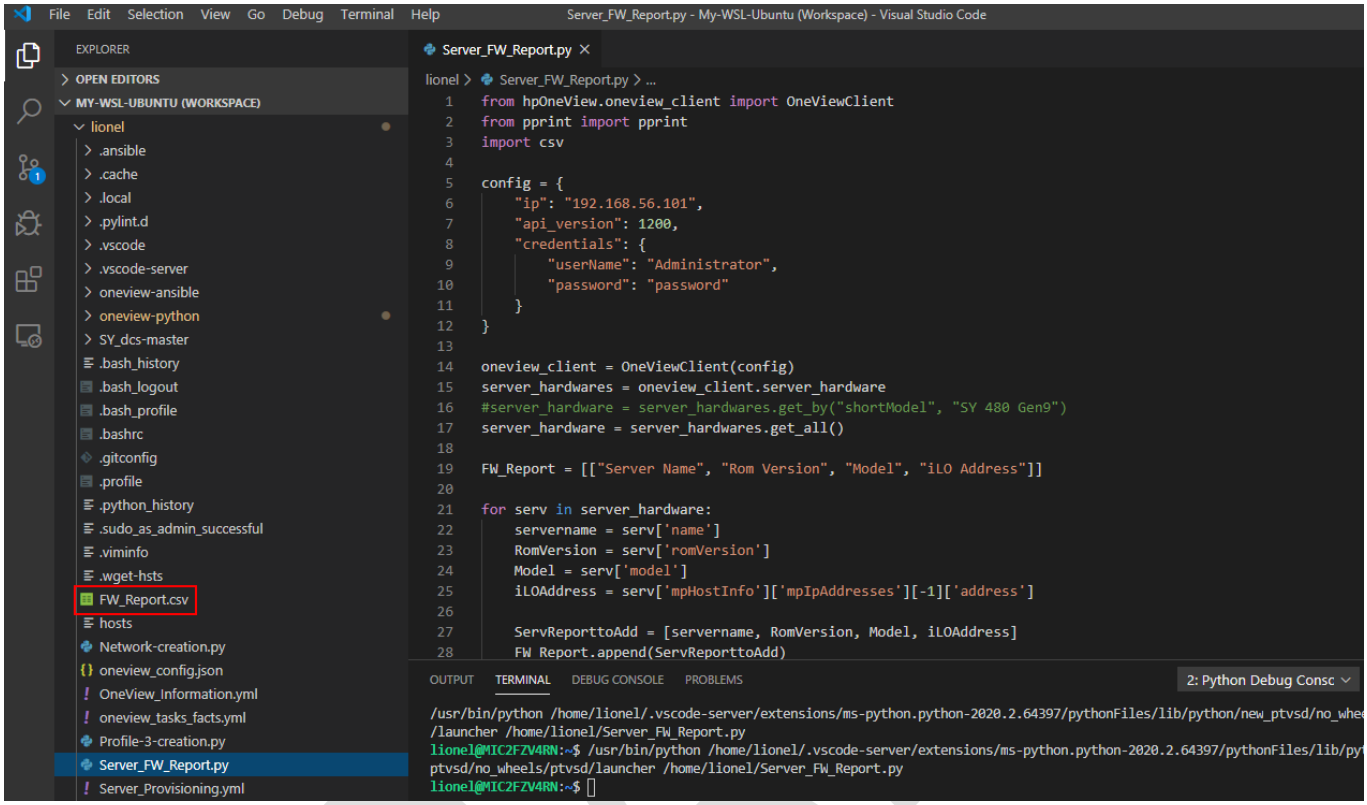


- Then add the following lines to your script:

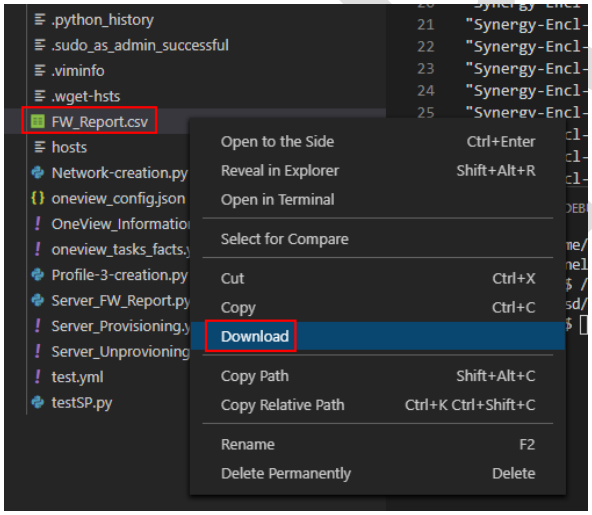
```
from hpOneView.oneview_client import OneViewClient
from pprint import pprint
import csv

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

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

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

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

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

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

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

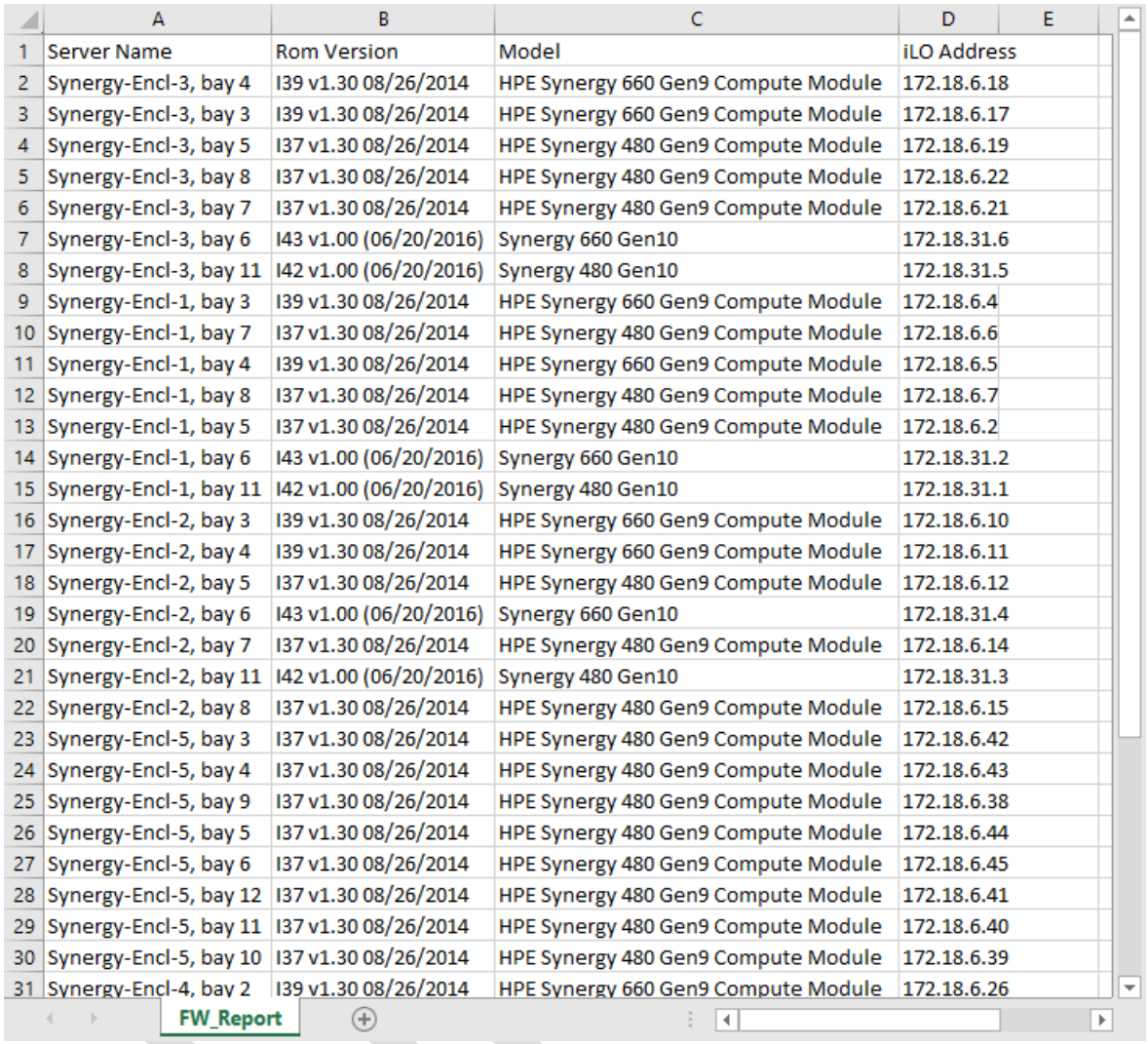
- Once run, you should find the **FW_Report.csv** file in your user home directory:



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



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



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

FW_Report

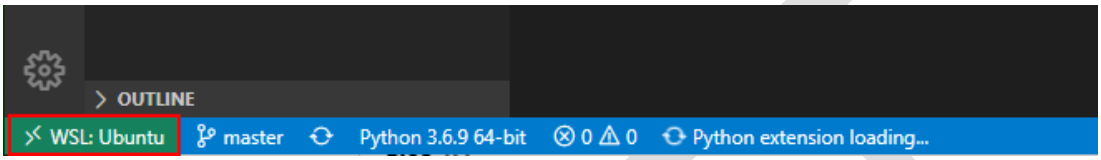
This concludes Python – Scenario 2

Python – Scenario 3 - Accelerating a configuration change

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

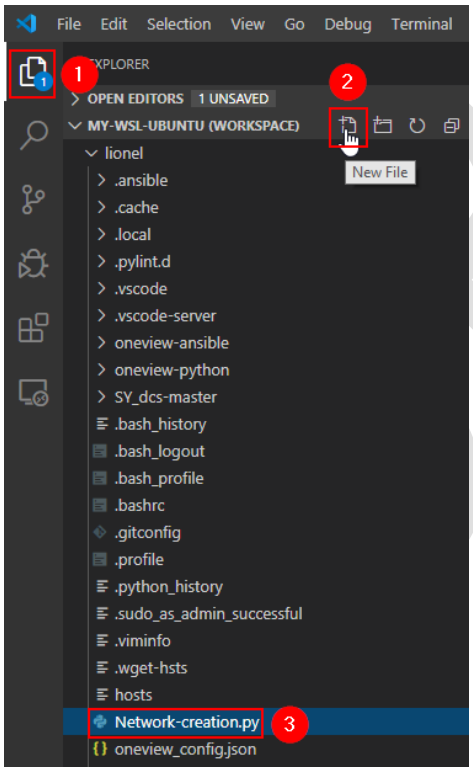
Prerequisites:

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



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

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



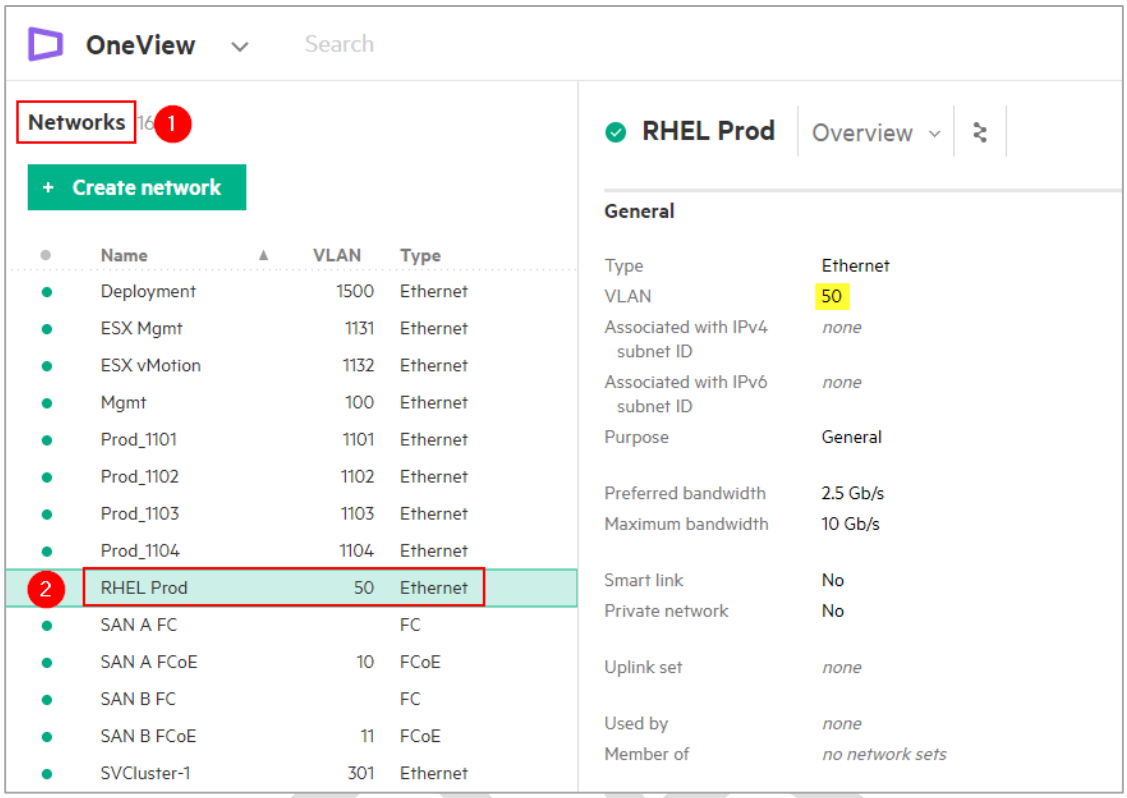
- Then add the following lines to your script:

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

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

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

oneview_client = OneViewClient(config)

ethernet_networks = oneview_client.ethernet_networks
network_sets = oneview_client.network_sets

# Creating Ethernet network

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

ethernet_network = ethernet_networks.create(options_ethernet)

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

# Get RHEL Prod network URI & name

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

logical_interconnect_groups  = oneview_client.logical_interconnect_groups

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

#help(oneview_client.logical_interconnect_groups)

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

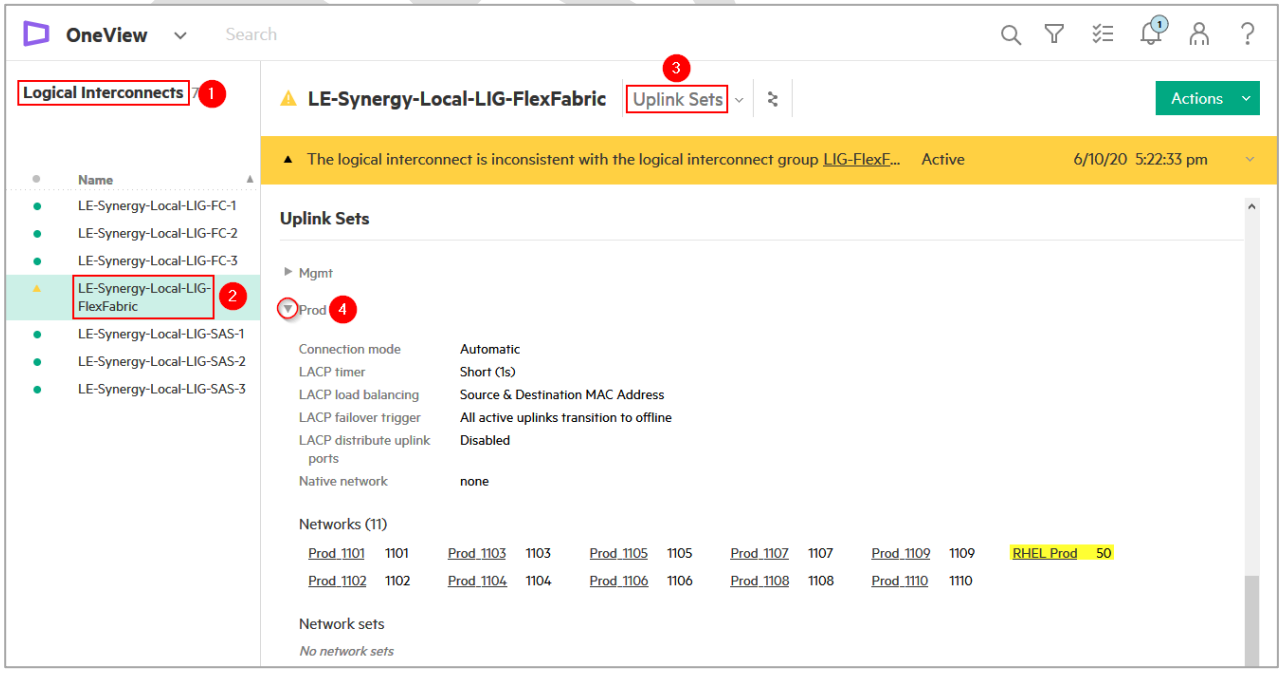
```
lig_to_update = lig_response.copy()

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

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

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

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


# Adding new network to Network Set
network_sets = oneview_client.network_sets
network_set = network_sets.get_by_name(networkset_name)
networkset_networkUris = (network_set.data)['networkUris']
new_networkset_networkUris = networkset_networkUris + [ethernet_network_uri]
network_set_update = {'networkUris': new_networkset_networkUris}
network_set = network_set.update(network_set_update)

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

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

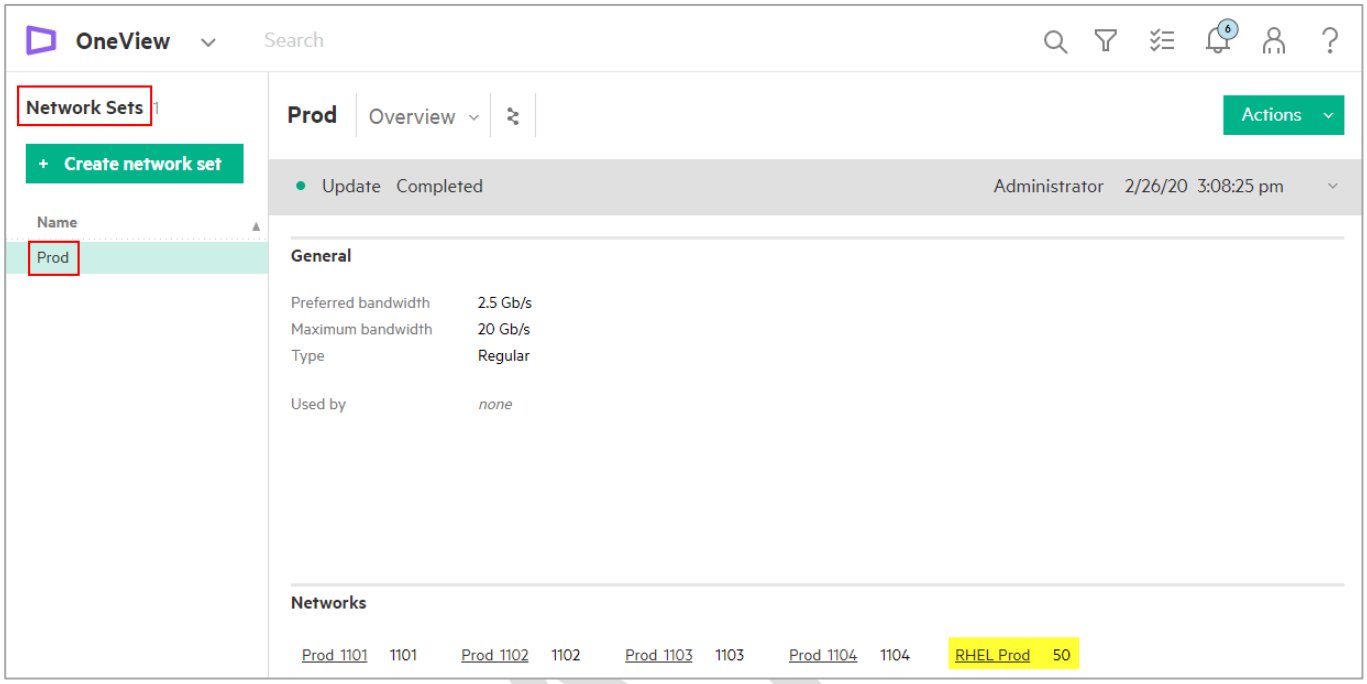
- Simultaneously, you can open the OneView web interface to show the script progress, open the **Networks** page to show the new **RHEL Prod** network:



- 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:



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



- 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

Ansible – Scenario 1 – Collecting facts in OneView

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

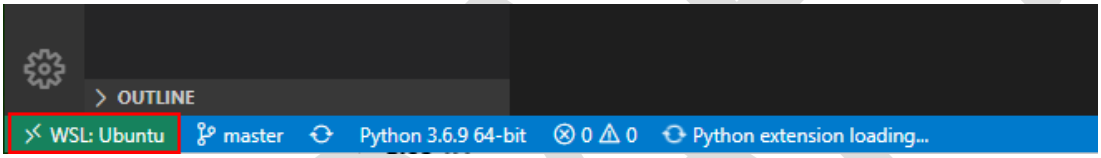
The integration of Ansible with HPE OneView extends the ability to automate the provisioning of bare-metal resources, including servers, storage, and networking. This accelerates time to value through automated, consistent provisioning.

> **Note**: To learn more about Ansible with HPE OneView, see Accelerating DevOps with HPE OneView and Ansible

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

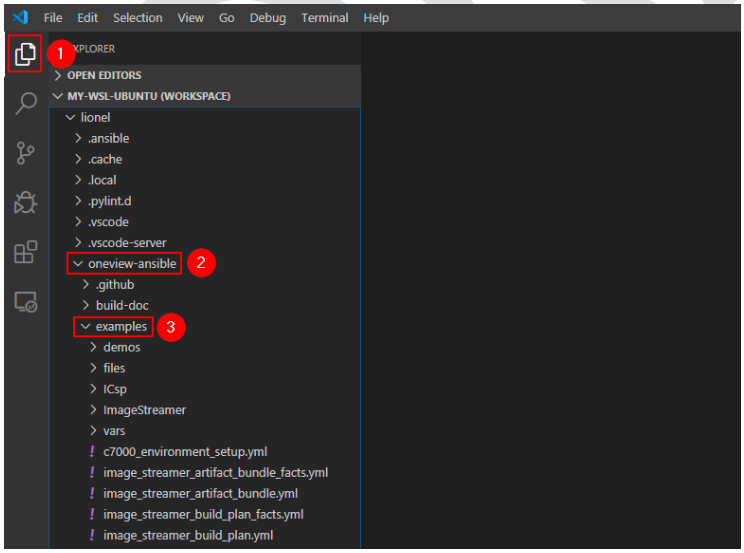
Prerequisites:

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



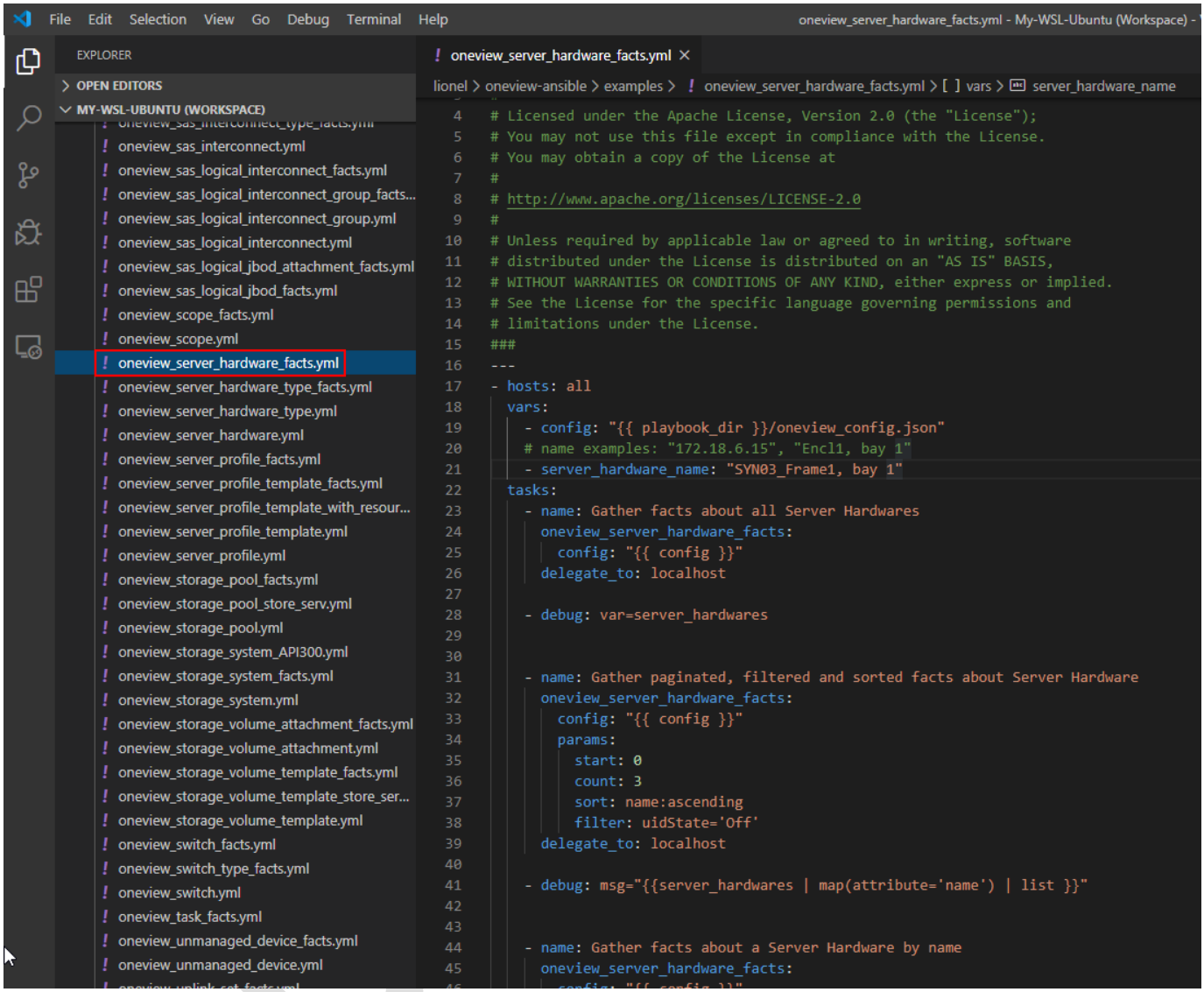
- **Notice**: This scenario can be run with either the initial or final configuration snapshot.

- In Explorer, open the **oneview-ansible** folder and browse the **examples** folder:

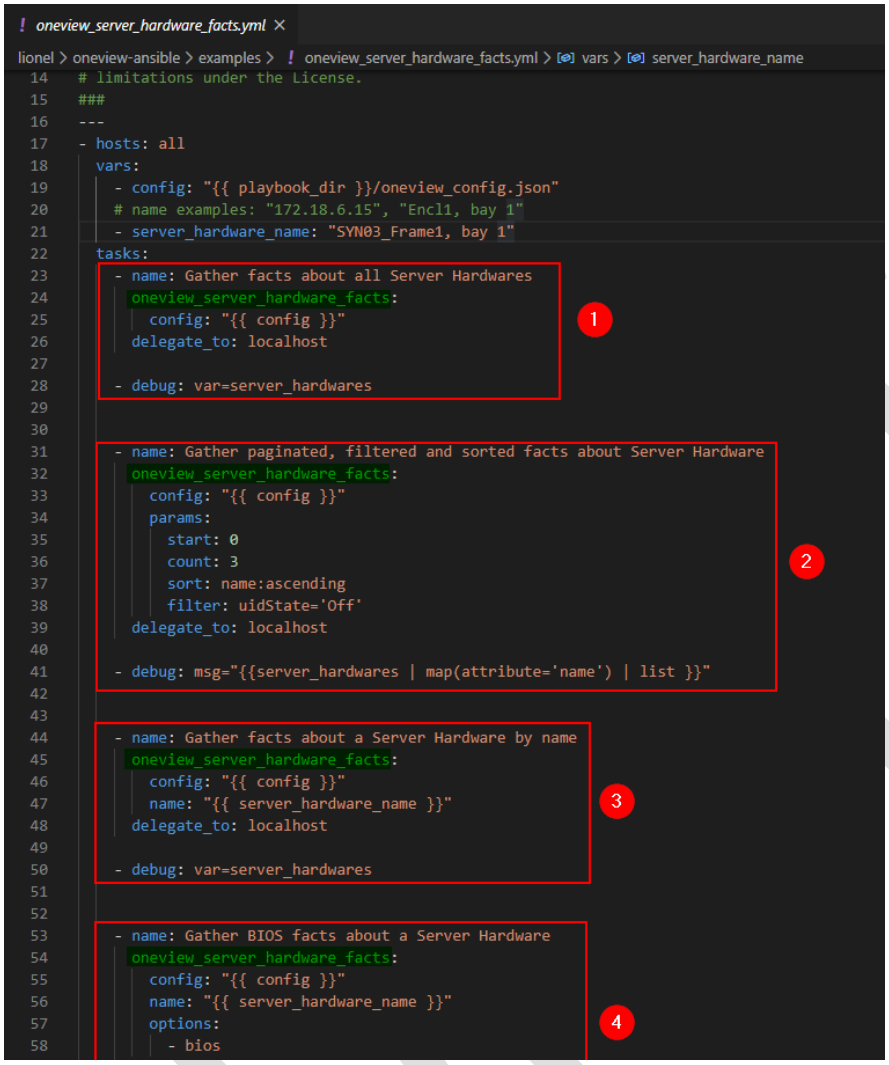


A long list of examples is provided by the *HPE OneView-Ansible* library. Each OneView resource operation is exposed through an Ansible module.

- Open **oneview_server_hardware_facts.yml** as an example:



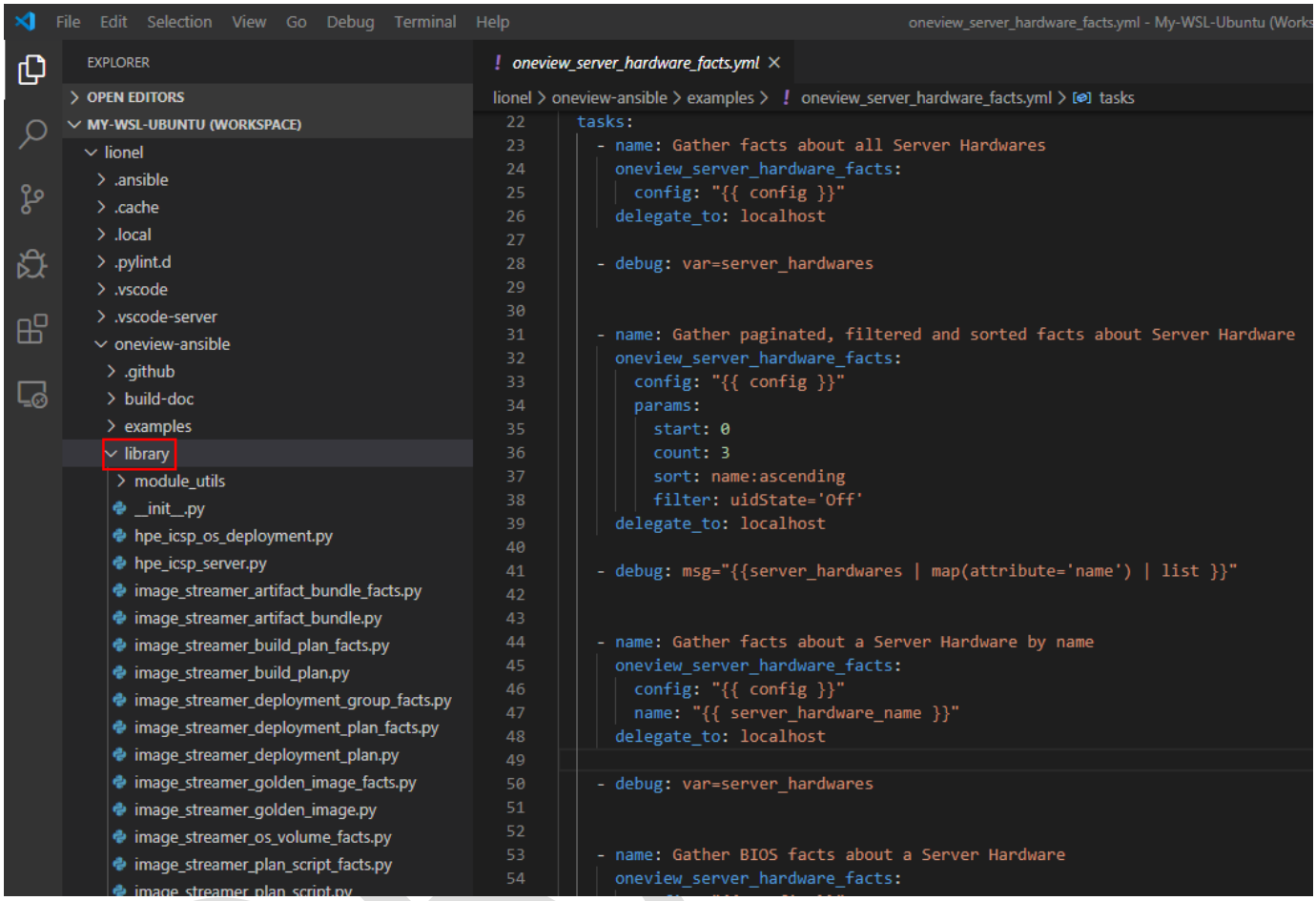
- This playbook provides several tasks examples on how to collect server hardware information. Each task uses the same `oneview_server_hardware_facts` Ansible module with different filter, options, and count parameters.



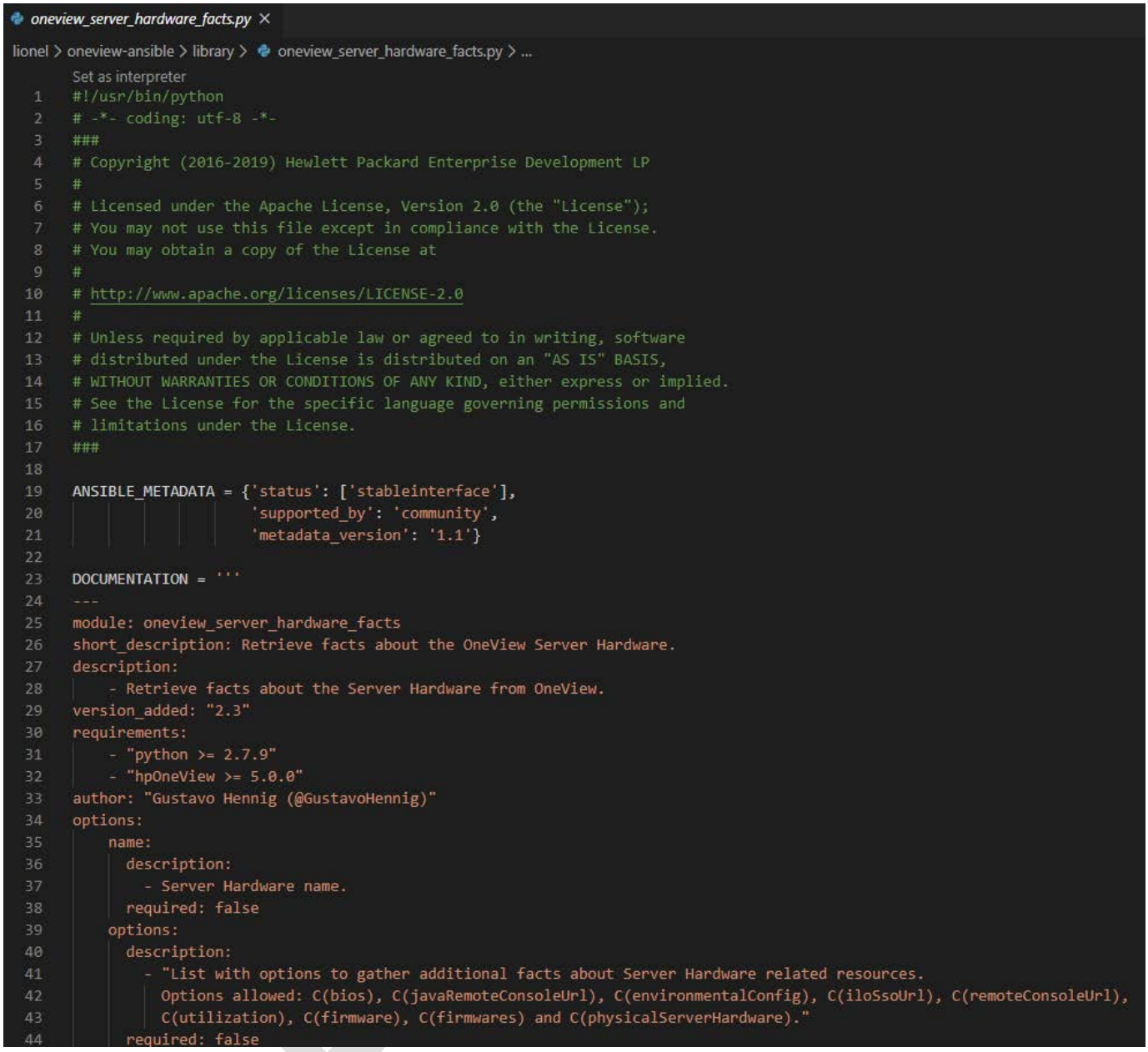
> **Note**: Ansible uses YAML syntax for their playbooks because it is easy for humans to read/write.

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

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



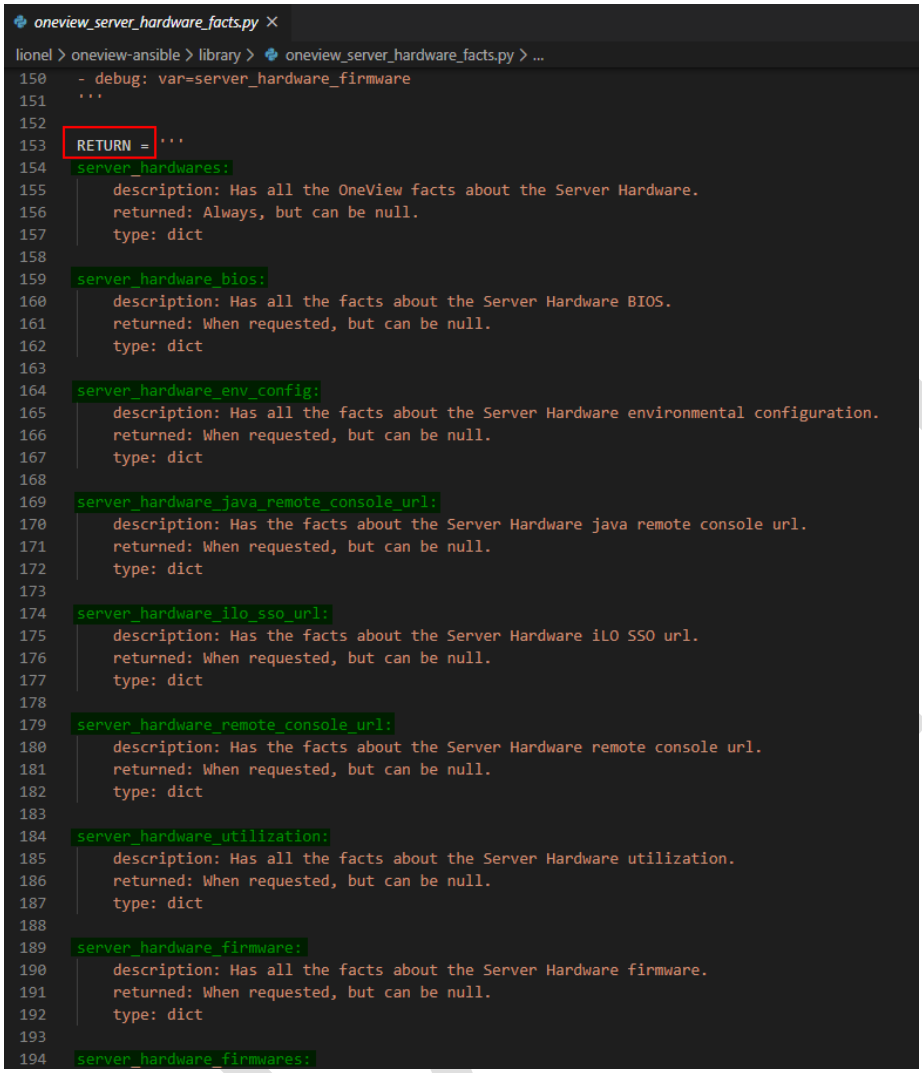
- Scroll-down and open the `oneview_server_hardware_facts.py` module:



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

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

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



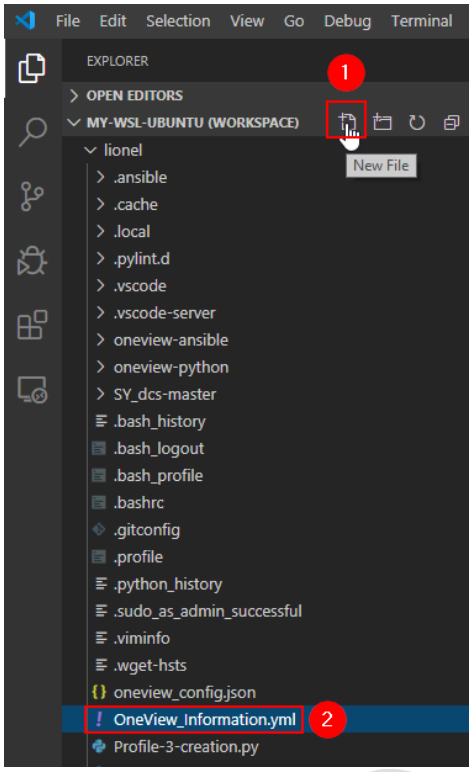
- As we can see, this module returns numerous values: `server_hardwares`, `server_hardware_bios`, `server_hardware_env_config`, etc. These are variables you can use in playbook to run additional tasks.

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

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

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

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

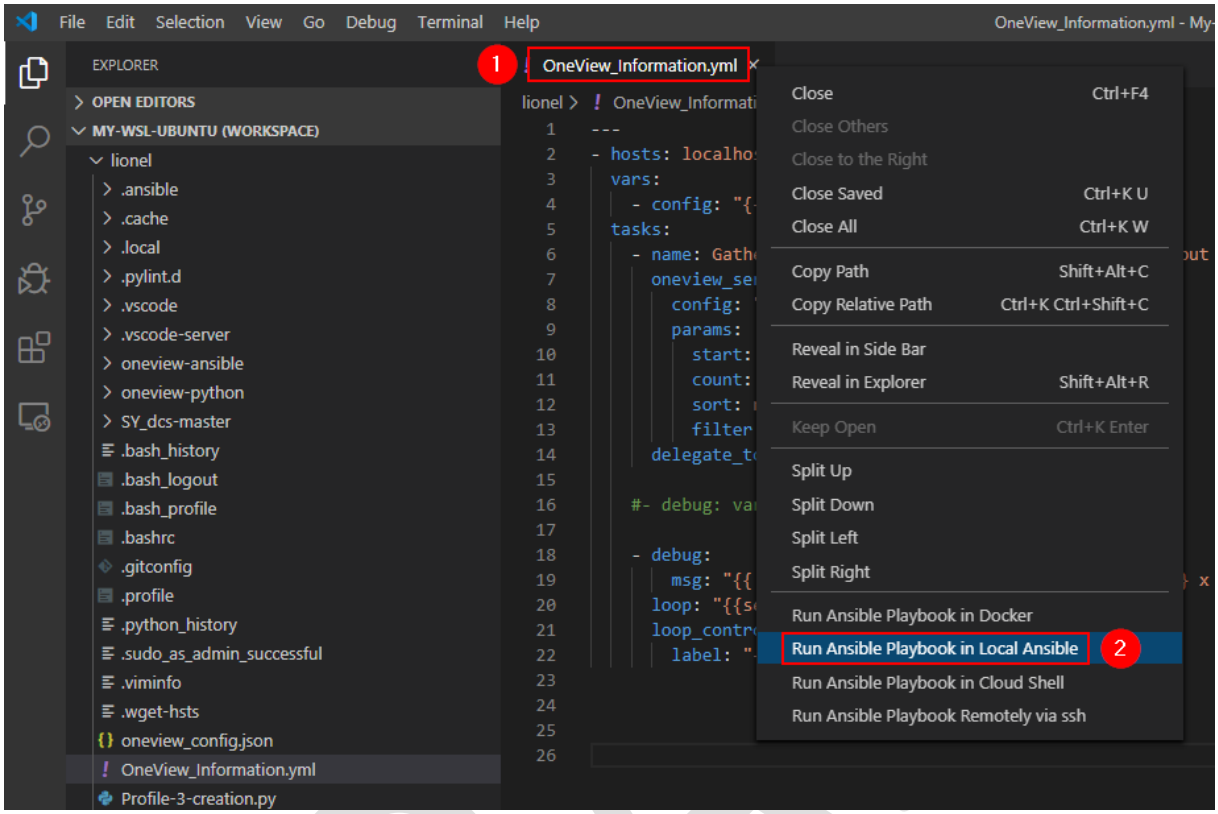


- Then copy/paste the following content:

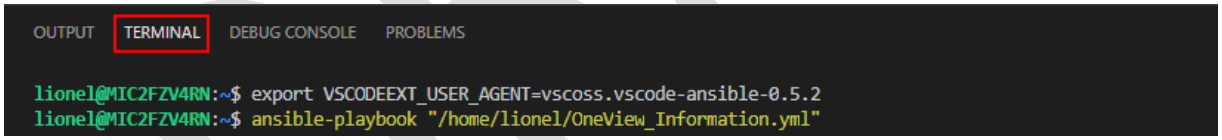
```
---
- hosts: localhost
  vars:
    - config: "{{ playbook_dir }}/oneview_config.json"
  tasks:
    - name: Gather paginated, filtered and sorted facts about Server Hardware
      oneview_server_hardware_facts:
        config: "{{ config }}"
        params:
          start: 0
          count: 3
 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

    #- debug: var=server_hardwares

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

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

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



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



- The console outputs the following:

```
PLAY [localhost] ***************************************************************************************************

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

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

TASK [debug] *******************************************************************************************************
ok: [localhost] => (item=Synergy 480 Gen10) => {
    "msg": "Synergy-Encl-1, bay 11 has 2 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}
ok: [localhost] => (item=HPE Synergy 660 Gen9 Compute Module) => {
    "msg": "Synergy-Encl-1, bay 3 has 4 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}
ok: [localhost] => (item=HPE Synergy 660 Gen9 Compute Module) => {
    "msg": "Synergy-Encl-1, bay 4 has 4 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz processors"
}

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

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

```
fatal: [localhost]: UNREACHABLE! => {"changed": false, "msg": "Failed to create temporary directory.In some c
ases, you may have been able to authenticate and did not have permissions on the target directory. Consider c
hanging the remote tmp path in ansible.cfg to a path rooted in \"/tmp\", for more error information use -vvv.
 Failed command was: ( umask 77 && mkdir -p \"` echo /home/[redacted]/.ansible/tmp `\"&& mkdir /home/[redacted]/.ansible/tm
p/ansible-tmp-1591538856.968808-12936-59109817527365 && echo ansible-tmp-1591538856.968808-12936-591098175273
65=\"` echo /home/[redacted]/.ansible/tmp/ansible-tmp-1591538856.968808-12936-59109817527365 `\" ), exited with resu
lt 1, stdout output: ansible-tmp-1591538856.968808-12936-59109817527365=/home/[redacted]/.ansible/tmp/ansible-tmp-15
91538856.968808-12936-59109817527365\n", "unreachable": true}
```

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

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

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

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

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

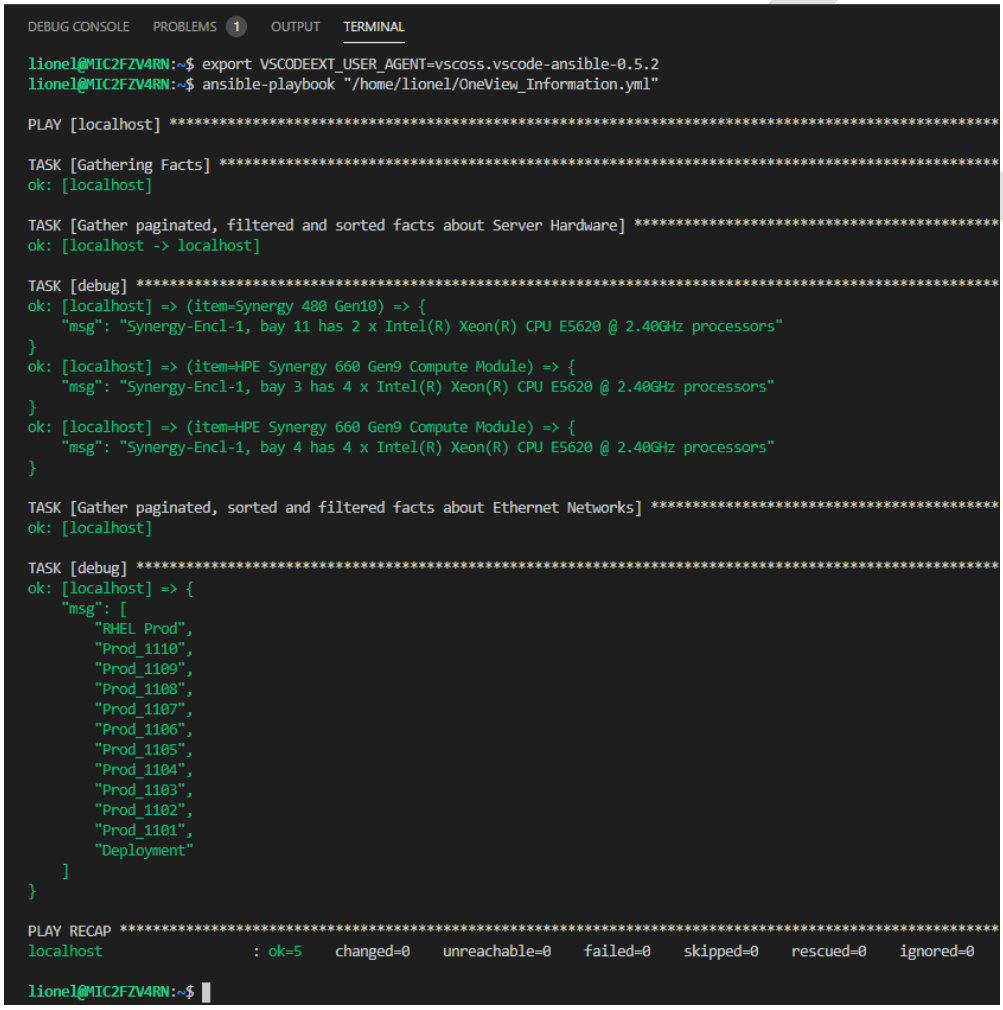
We use the `loop` option to go through the content of `server_hardwares` a dictionary object.

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

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

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

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

- Save the file then run it:



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

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

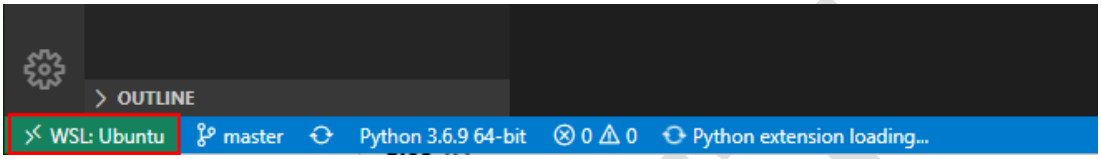
This concludes Ansible – Scenario 1 about collecting facts in OneView

Ansible – Scenario 2 – Provisioning New Servers

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

Prerequisites:

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



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

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

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

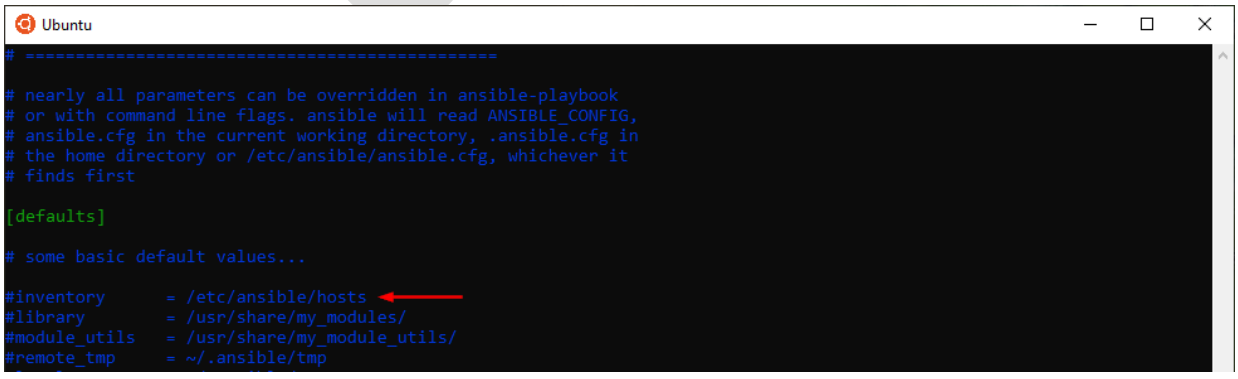
To illustrate this, we are going to create a *RHEL* group in the *hosts* file with two systems, *RH-1* and *RH-2*.

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

- From the Ubuntu console and enter:

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

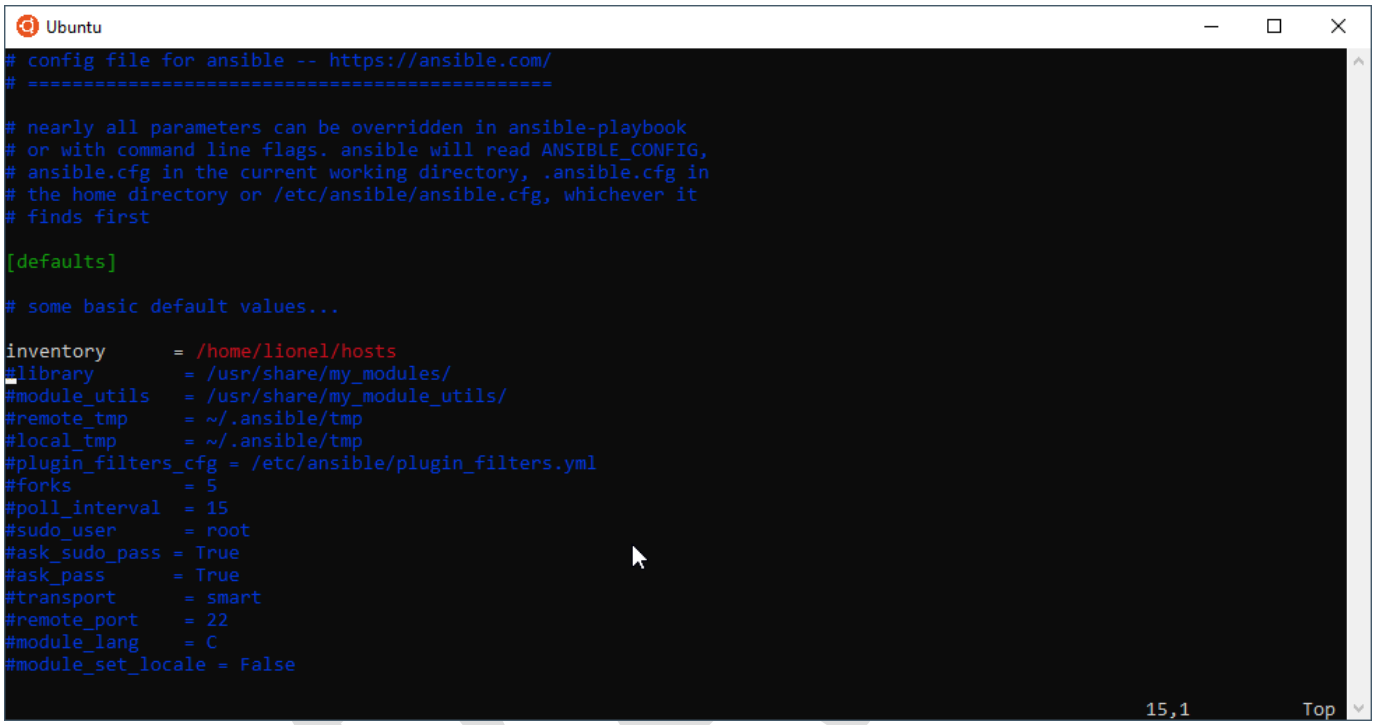
- Enter your password.
- Uncomment the inventory value:



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

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

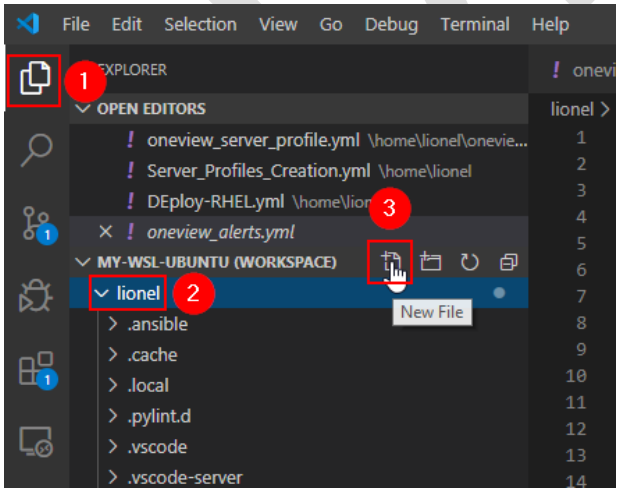
with <username> your username



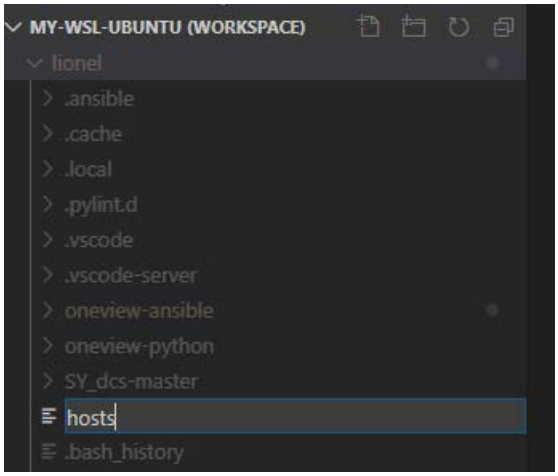
- Press **ESC** to exit *Insert* Mode, then type **:** (colon) to open the vi command prompt, type **wq** and press **ENTER** to Write and Quit vi

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

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



- Named it **hosts**:

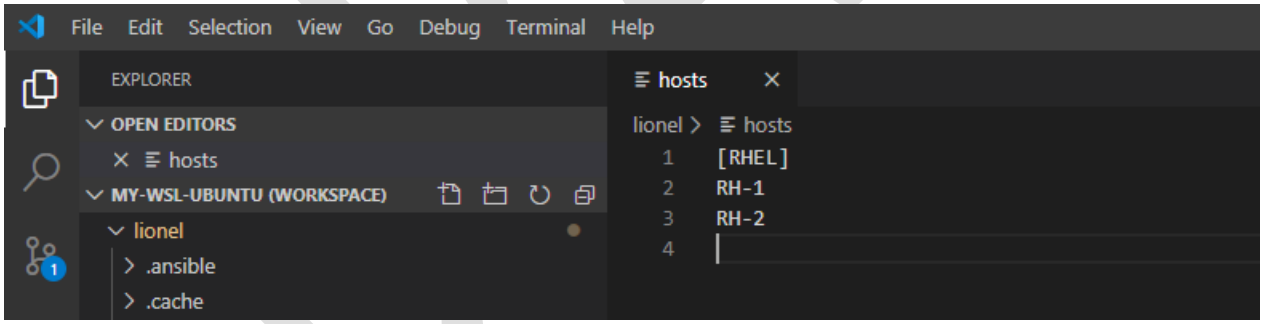


- Then add in the hosts file the following content:

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

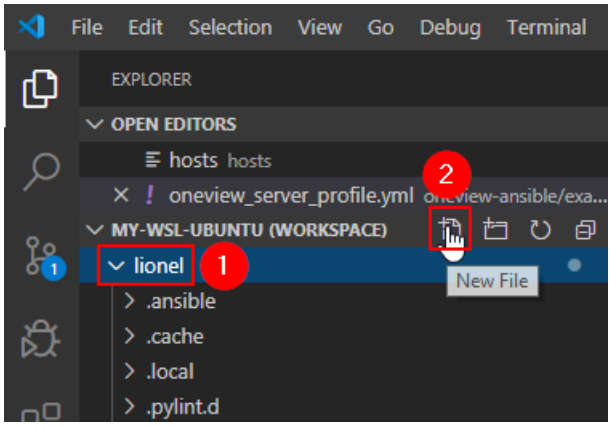
- Then save the file by pressing **CTRL** + **S**

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

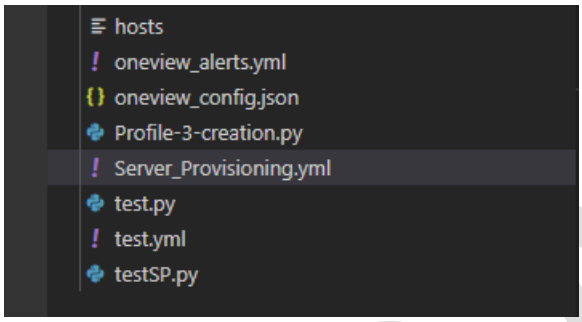


Now let's build a new playbook that will use this *hosts* file to create two server profiles (RH-1 and RH-2) from an existing Server Profile Template.

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



- Enter the name **Server_Provisioning.yml**



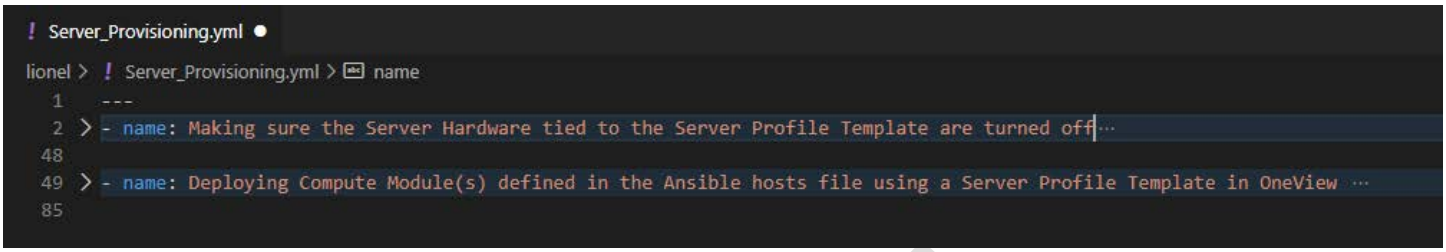
- Copy/paste the following playbook content:

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

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

    # - debug:  var=server_profile_templates

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

    # - debug:  var=serverHardwareTypeUri

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

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

    # - debug: var=server_hardware_names

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

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

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

    #- debug: var=server_hardware

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

  - name: Powering on the Compute Module(s) [{{ server_hardware.name }}]
    oneview_server_hardware:
        config: "{{ config }}"
        state: power_state_set
        data:
            name : "{{ server_hardware.name }}"
            powerStateData:
                powerState: "On"
                powerControl: "MomentaryPress"
    delegate_to: localhost

  - debug:
        msg: "The server is located in {{ server_hardware.name }}"
```

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

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



In the first group of tasks, we make sure the server hardware that are using the same server hardware type as the server profile template are turned off so that we can create the server profiles. Notice we do not run these tasks using the *hosts* file but instead using *localhost*.

- ❶: We gather the facts about the server profile template we use.
- ❷: We set a variable to collect the server hardware type uri used by the server profile template we use.
- ❸: We gather the facts about the server hardware facts.
- ❹: We set a variable to collect the server hardware names that are using the server hardware type uri collected in ❷
- ❺: We turn off all server hardware using the names collected in ❹

Server_Provisioning.yml

lionel > Server_Provisioning.yml > name

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

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

    # - debug:  var=server_profile_templates

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

    # - debug:  var=serverHardwareTypeUri

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

    # - debug:
    #     msg: "{{ server_hardwares  | selectattr('serverHardwareTypeUri', 'equalto', serverHardwareTypeUri )

    - set_fact:
        server_hardware_names : "{{ server_hardwares  | selectattr('serverHardwareTypeUri', 'equalto', serverHa

    # - debug: var=server_hardware_names

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

- name: Deploying Compute Module(s) defined in the Ansible hosts file using a Server Profile Template in OneVi
```

1 2 3 4 5

In the second group of tasks, we create the server profiles. Notice that this time, we run those tasks using the *hosts* file:

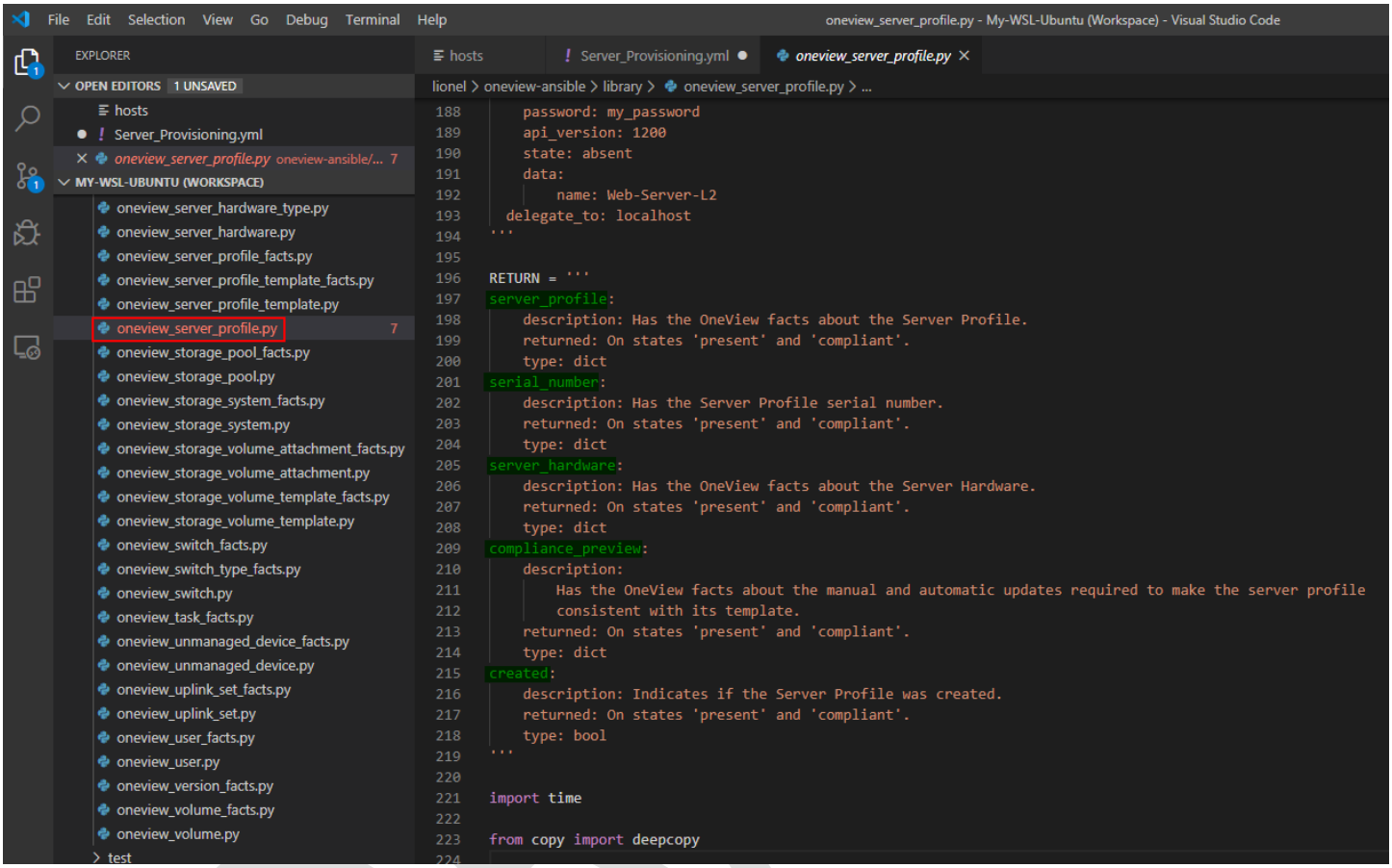
- ❶: We create one or more server profiles according to the *RHEL* group settings found in the hosts inventory file. The group is set by `hosts: RHEL` at the beginning of the second group of tasks.
- ❷: We defines the Server Profile Template that Ansible must use to generate the Server Profiles, the Server Profile Template is set in line 54.
- ❸: We tell Ansible to use the hosts inventory file to generate the Server Profile names.
- ❹: We display the result of the Server Profile creation task

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

  tasks:
    - name: Creating Server Profile [{{ inventory_hostname }}] from Server Profile Template [{{ server_template }}]
      oneview_server_profile:                                                                                          ← 1
        config: "{{ config }}"
        data:
          serverProfileTemplateName: "{{ server_template }}"                                                           ← 2
          name: "{{ inventory_hostname }}"                                                                             ← 3
      delegate_to: localhost
      register: result

    #- debug: var=server_hardware

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

Then in the next subtasks, the servers are powered-on in ❶ then we display in ❷ the server location.

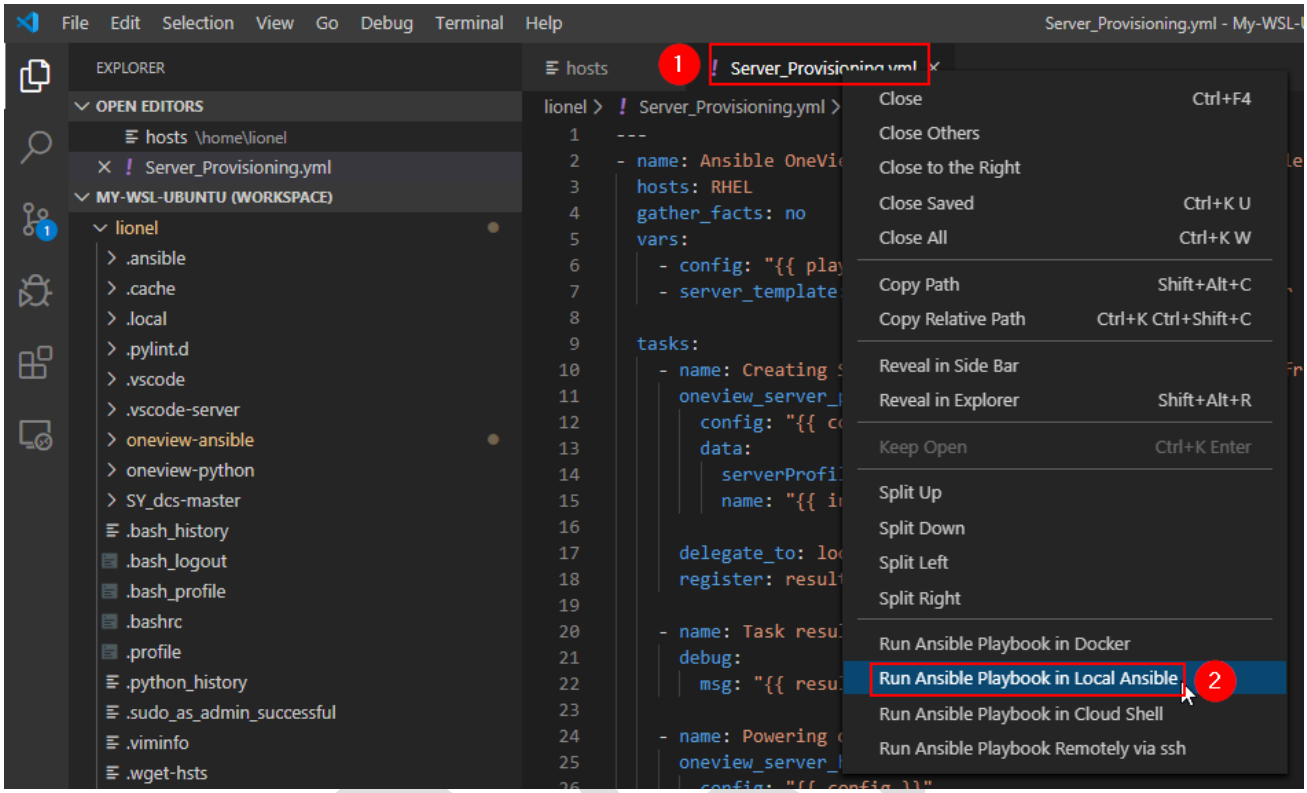
```
    - name: Powering on the Compute Module(s) [{{ server_hardware.name }}]                                             ← 1
      oneview_server_hardware:
        config: "{{ config }}"
        state: power_state_set
        data:
            name : "{{ server_hardware.name }}"
            powerStateData:
                powerState: "On"
                powerControl: "MomentaryPress"
      delegate_to: localhost

    - debug:
        msg: "The server is located in {{ server_hardware.name }}"                                                     ← 2
```

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

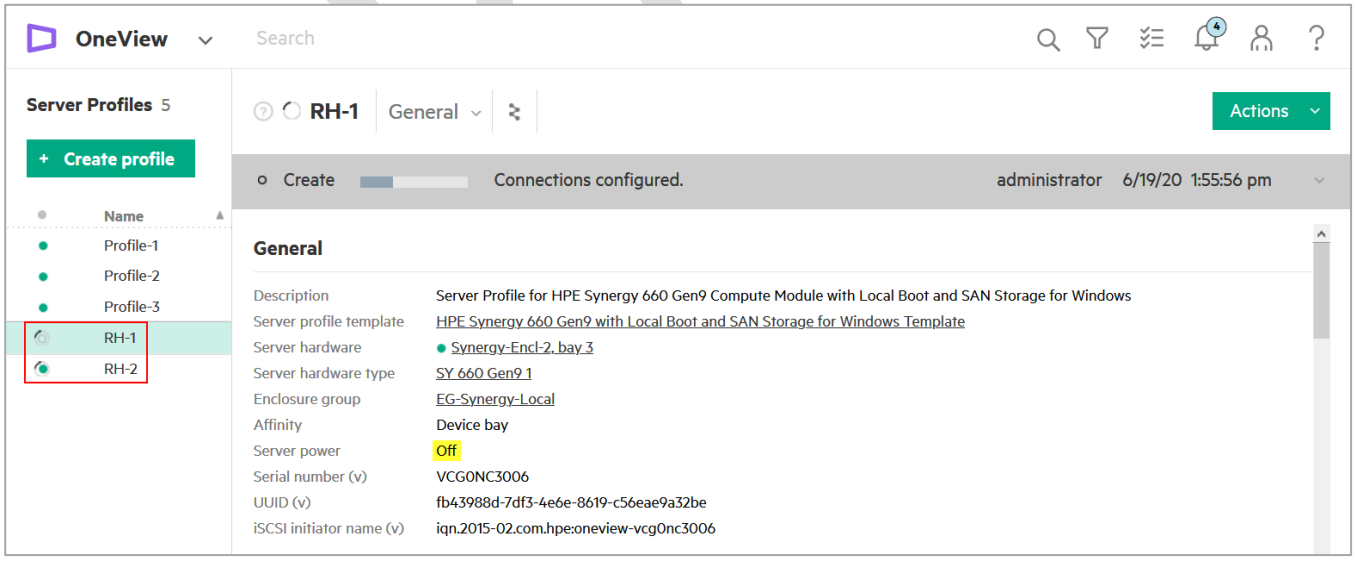


In our playbook, the last message we send to the console, is using one of those variables returned by the module when executed (i.e. `server_hardware`). We display only the *name* attribute using `server_hardware.name` to get only the server hardware location.

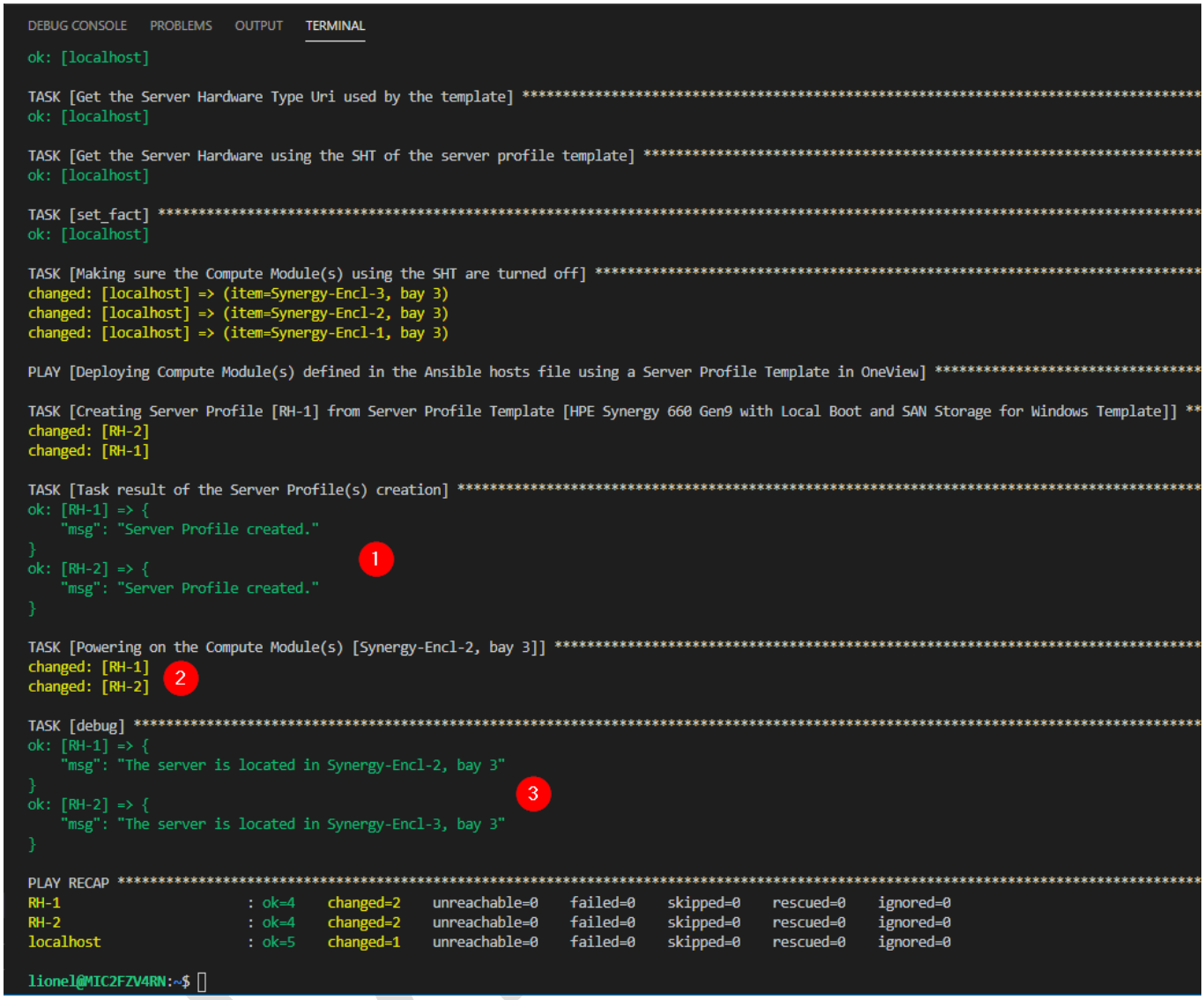
- Now to run the Ansible playbook, right-click on the tab then select **Run Ansible Playbook in Local Ansible**



- Open the OneView UI (https://192.168.56.101/#/profiles) to show the creation of the two server profiles:

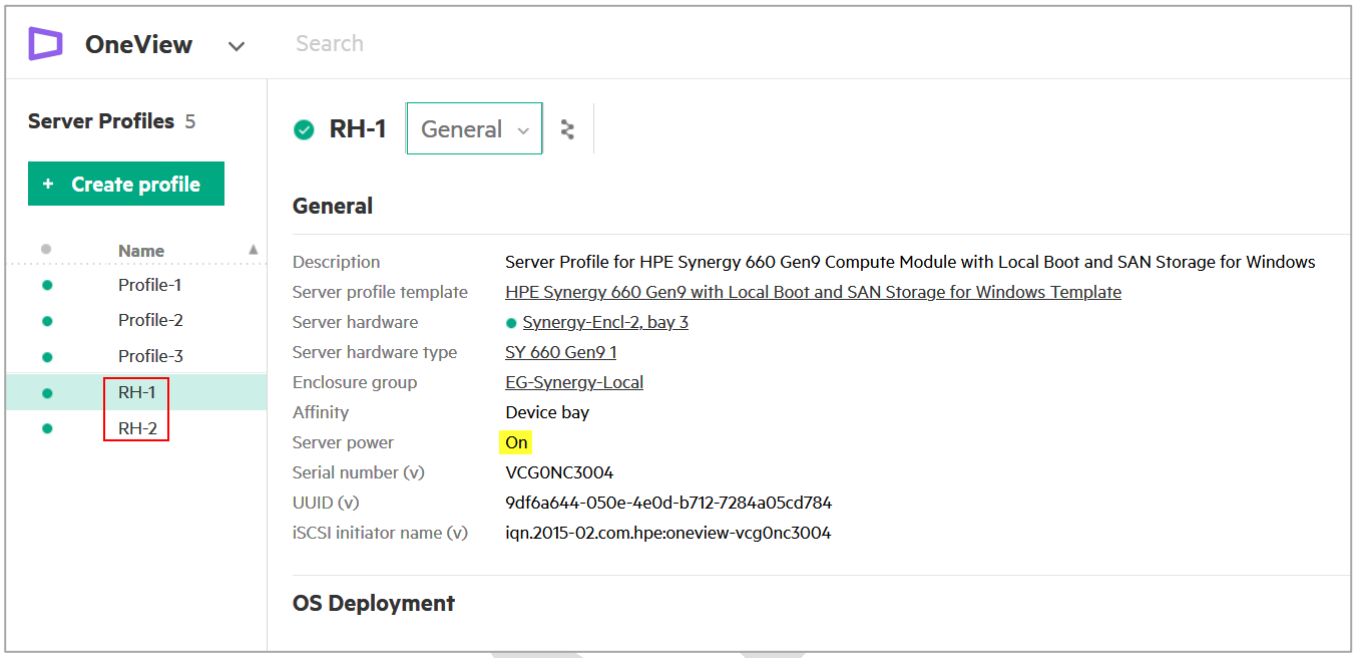


- Open the VS Code terminal to see the outputs of our Ansible playbook tasks:



- ❶: Task result of the server profiles creation.
- ❷: Task result of the powering-on of the server hardware.
- ❸: Displays the server hardware location used by the server profiles

- Once completed, two Server Profiles RH-1 and RH-2 are created, and the servers are powered-on:



This concludes Ansible - Scenario 2

Ansible – Scenario 3 – Unprovisioning Running Servers

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

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

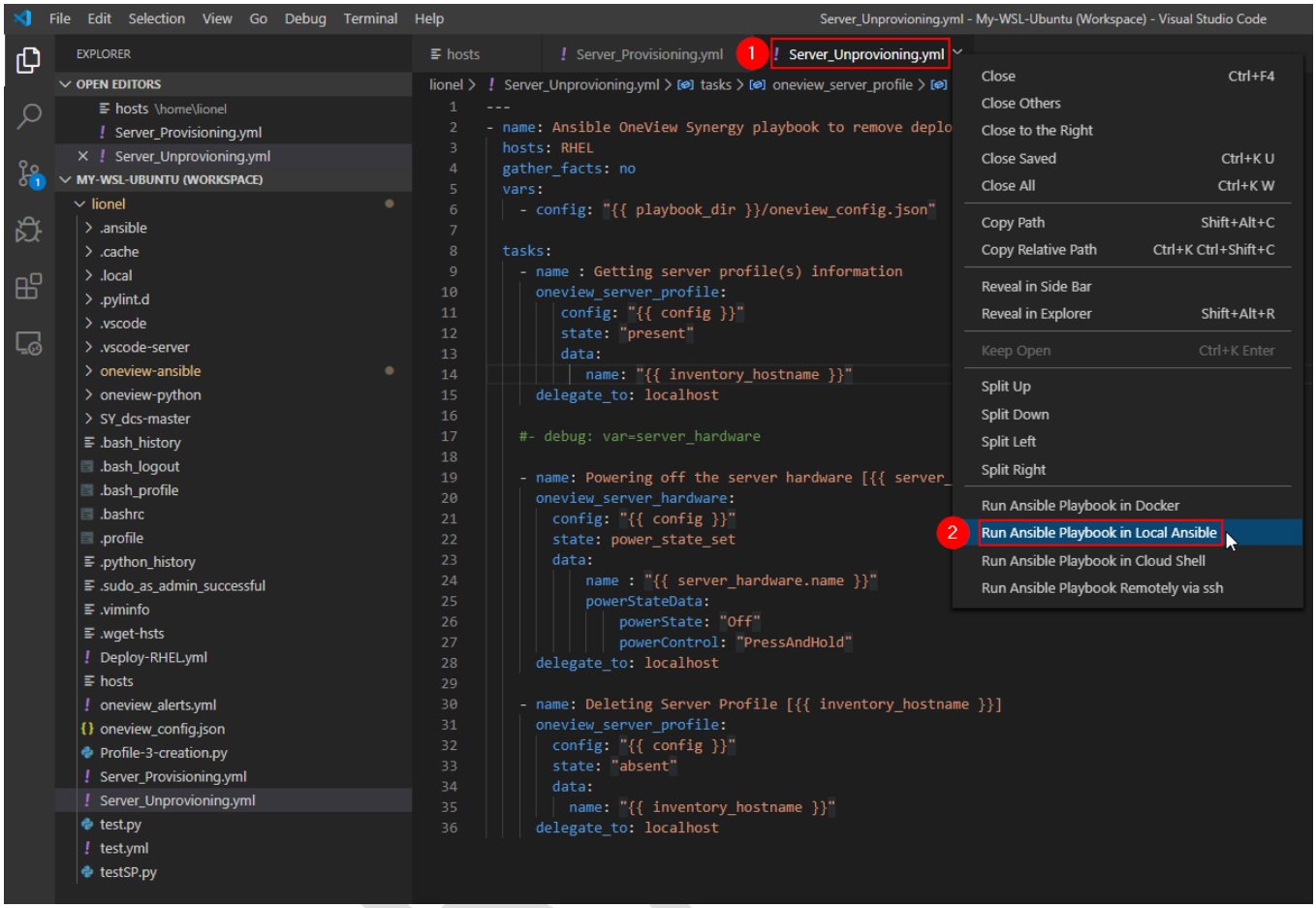
- To do this, create a new file, named it **Server_Unprovioning.yml** and add the following content:

```
---
- 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
```

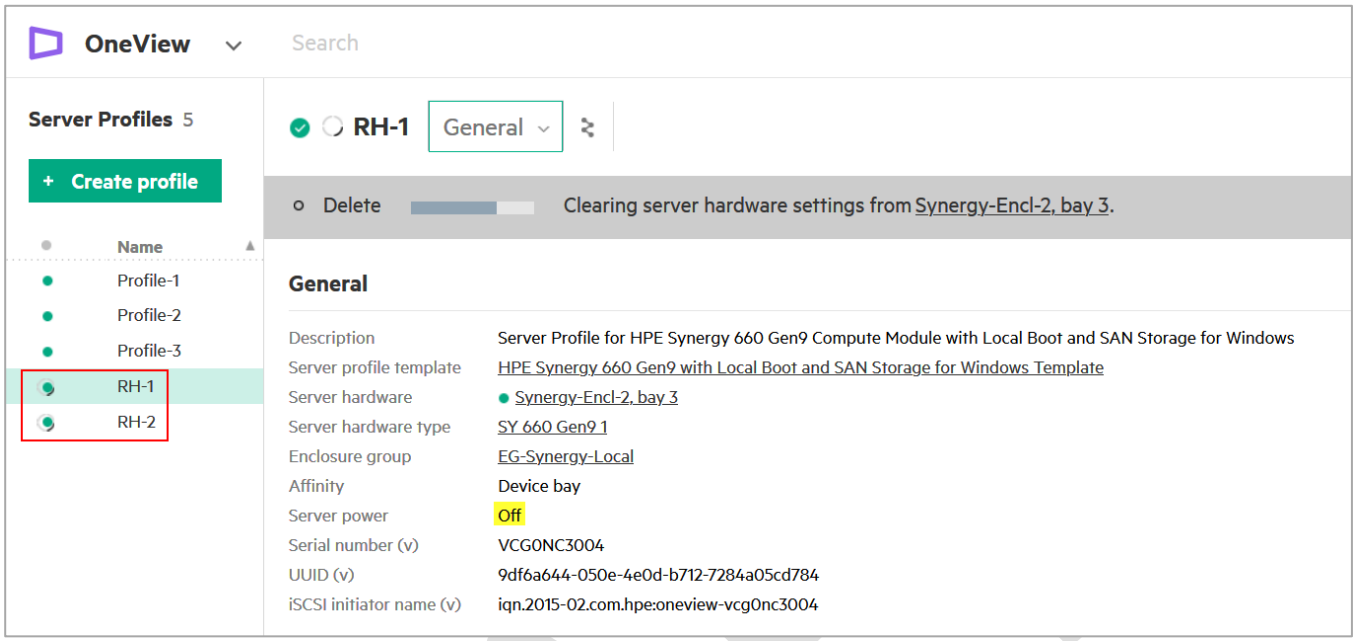
This playbook has three tasks:

1. Getting server profiles information using the *hosts* file inventory information.
2. Powering off the server hardware using the name collected from the server profile
3. Deleting the server profiles using the name pulled from the *hosts* files.

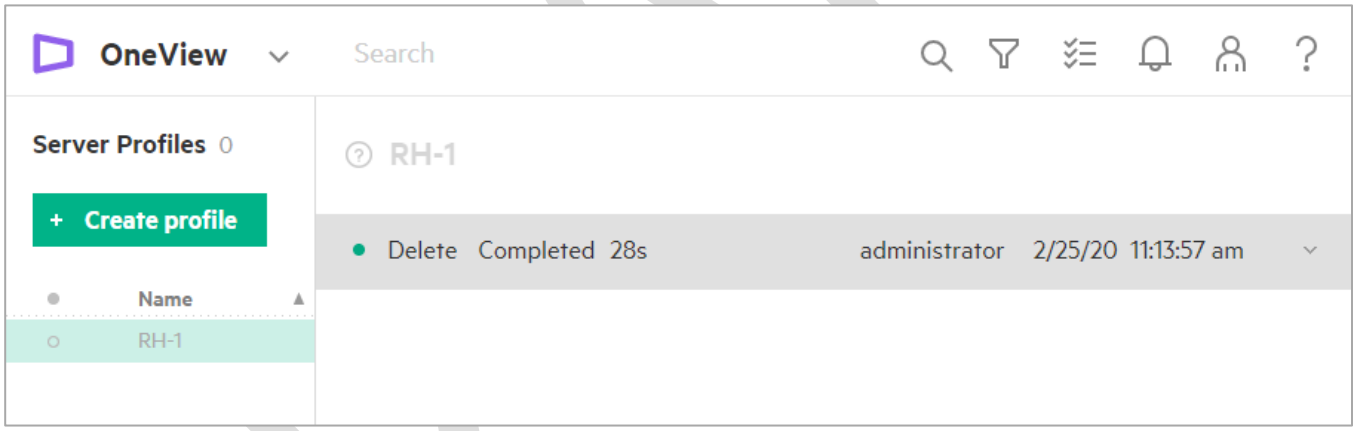
- To run the playbook, right click on the file tab then select **Run Ansible Playbook in Local Ansible**



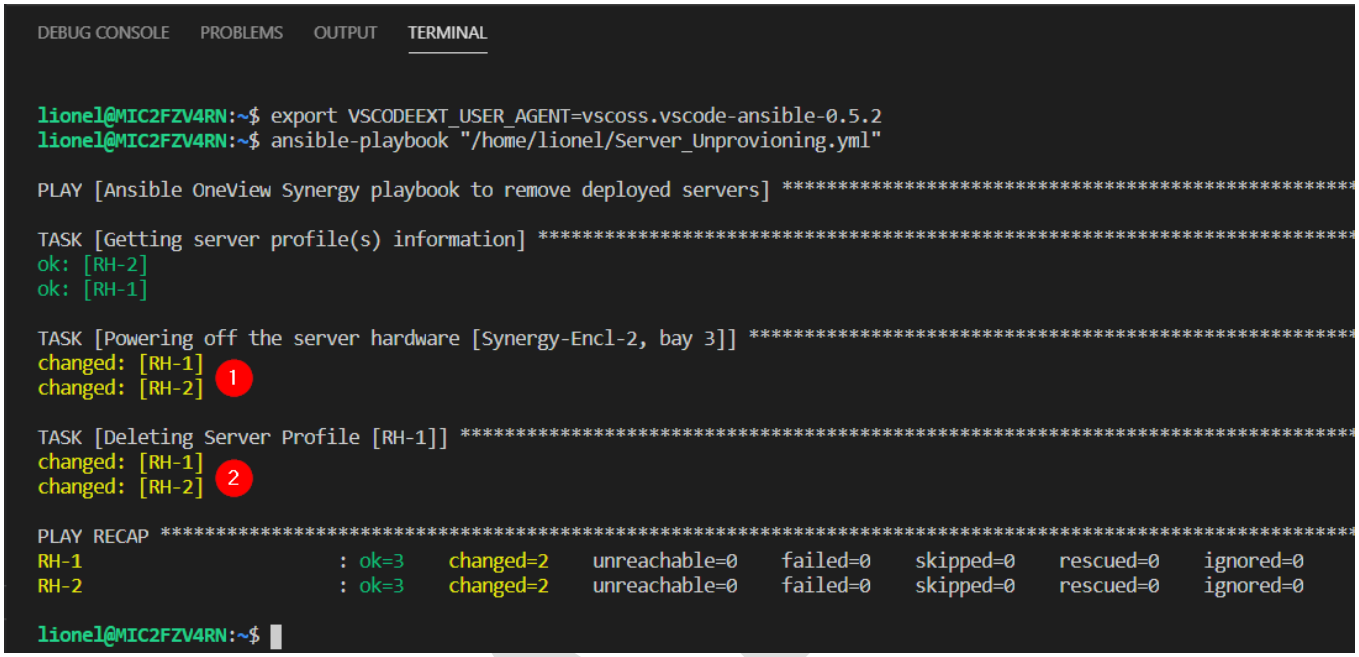
- Both servers should power-off and the server profiles should be erased.



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



- In the terminal, the following outputs get displayed:



- ❶: Task result of the server hardware powering-off.
- ❷: Task result of the server profiles deletion.

This concludes Ansible – Scenario 3

Terraform – Scenario 1 – Excution plan to accelerate a configuration change

Terraform, by HashiCorp, is a popular tool for infrastructure automation that facilitates a multi-cloud or hybrid cloud approach to infrastructure management. When integrated with HPE OneView, Terraform is a powerful orchestration tool that can be used to create, manage, and update infrastructure resources. These resources may be server profiles, server hardware, logical enclosure, networks, and so on.

Terraform is a tool not only for building and changing but also for versioning infrastructure. Configuration files that describe actions over components are used to generate an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is also able to determine what changed and create incremental execution plans which can be applied.

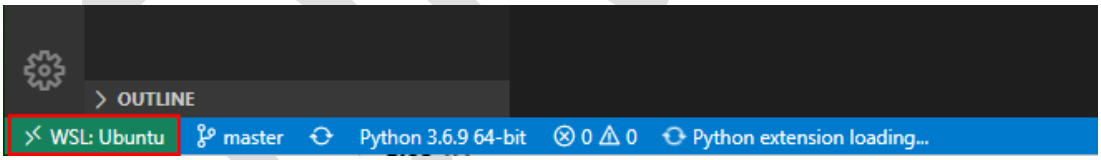
HPE OneView Terraform provider automates the provisioning of physical infrastructure on-demand using Software-Defined templates from HPE OneView. This enables administrators to create a resource topology similar to that of a public cloud on their own physical infrastructure. With this type of resource topology, administrators can easily migrate applications and workloads to an on-premises private cloud environment to realize their hybrid cloud strategy.

> **Note**: To learn more about Terraform with HPE OneView, see https://github.com/HewlettPackard/terraform-provider-oneview

In this scenario, we are going to use Terraform to add a new network and assign this network to any of the existing Server Profiles resources using the network set *Prod*.

Prerequisites:

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

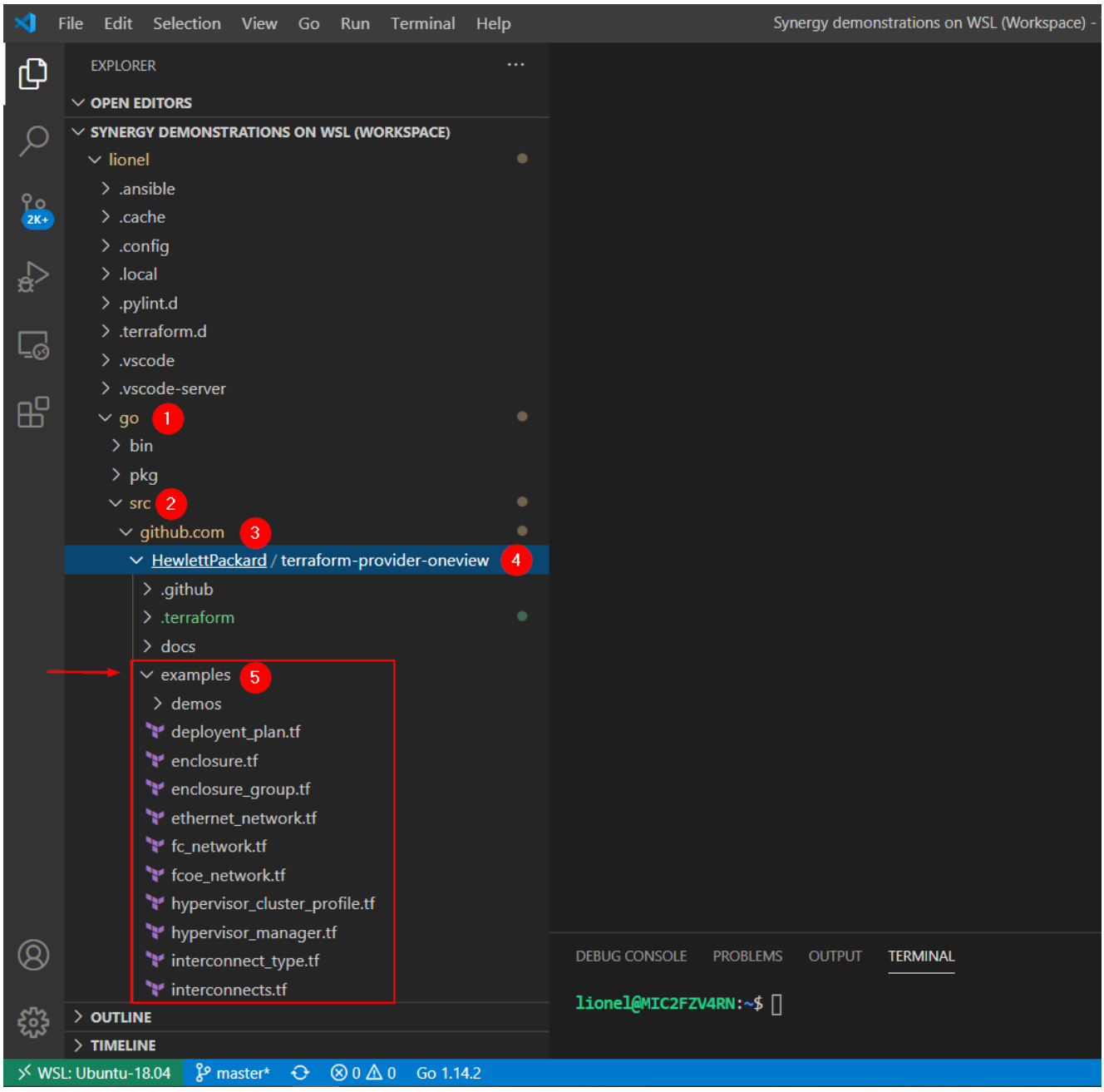


- **Important notice**: This scenario can only be run with the Final configuration snapshot (i.e. when a Logical Enclosure is available)

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

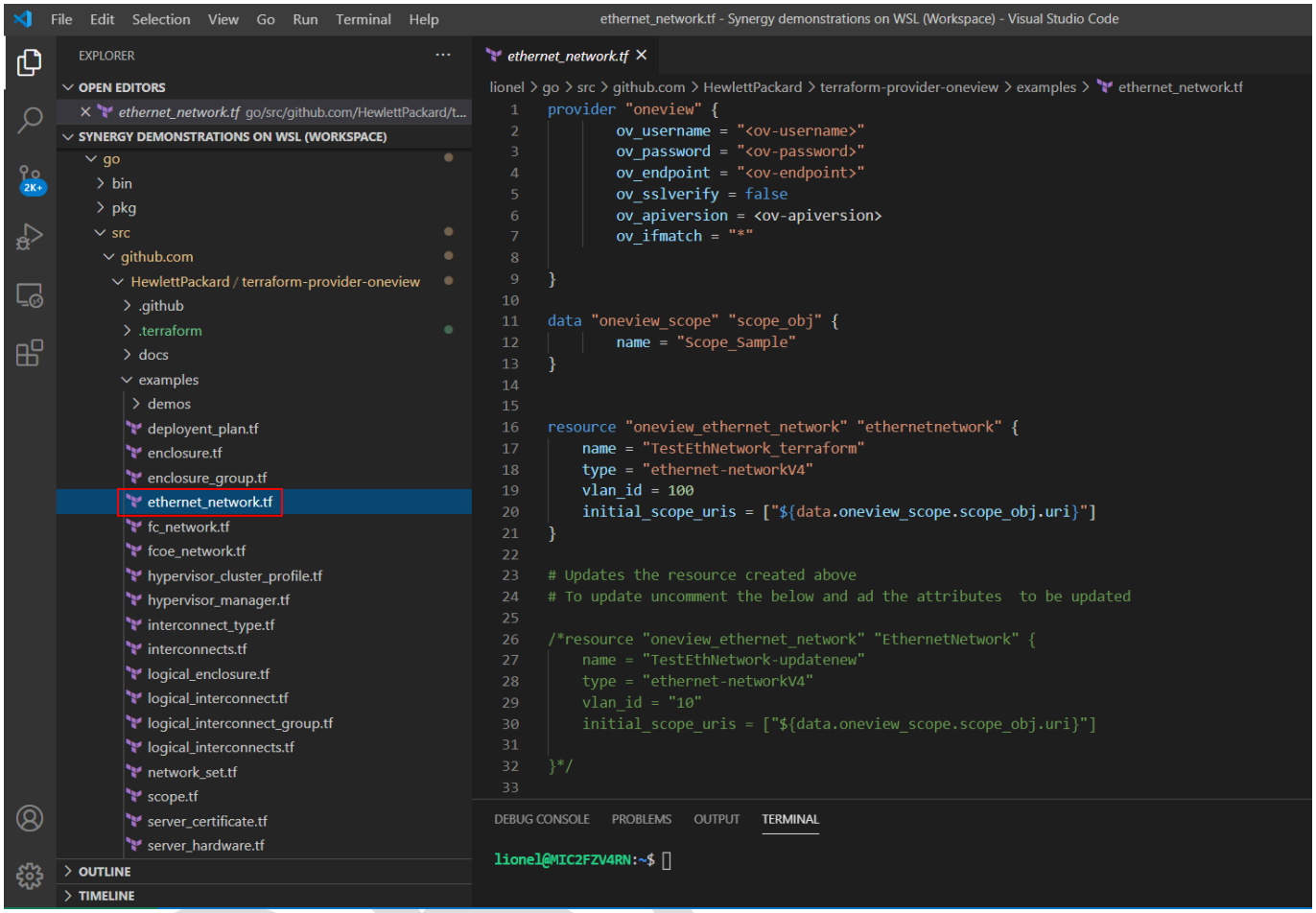
To access the Terraform examples:

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



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

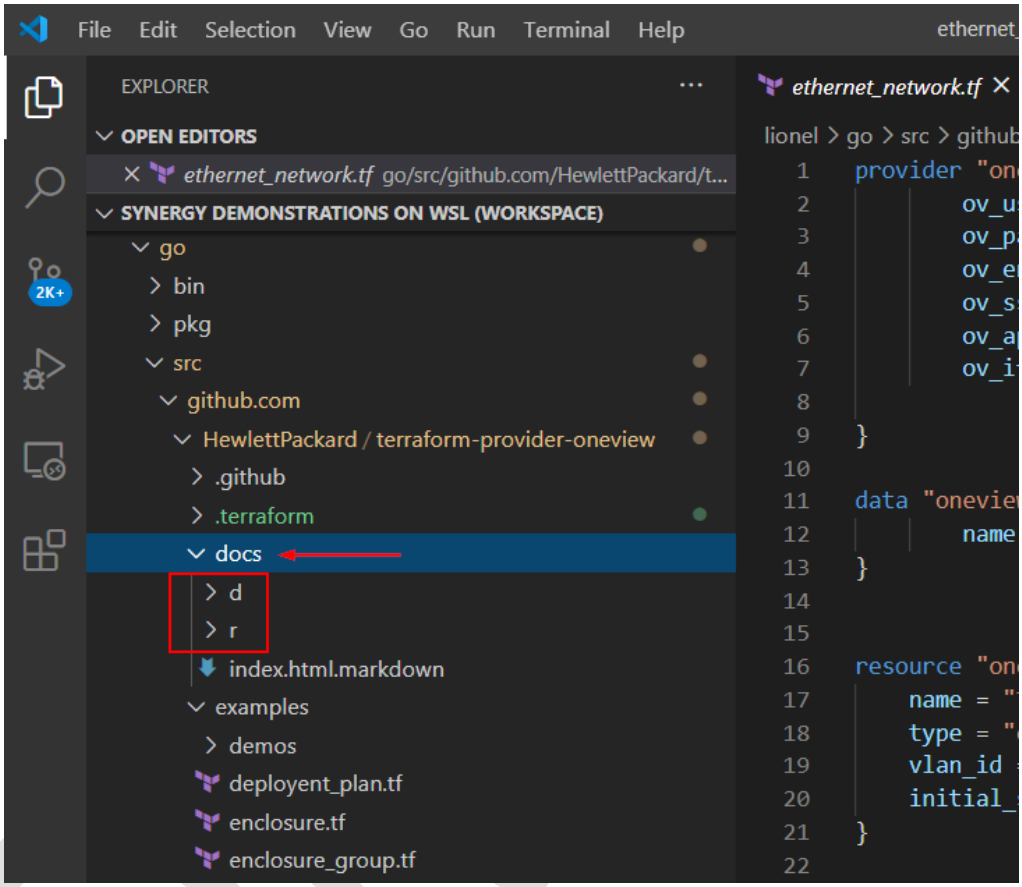
- Select **ethernet_network.tf**



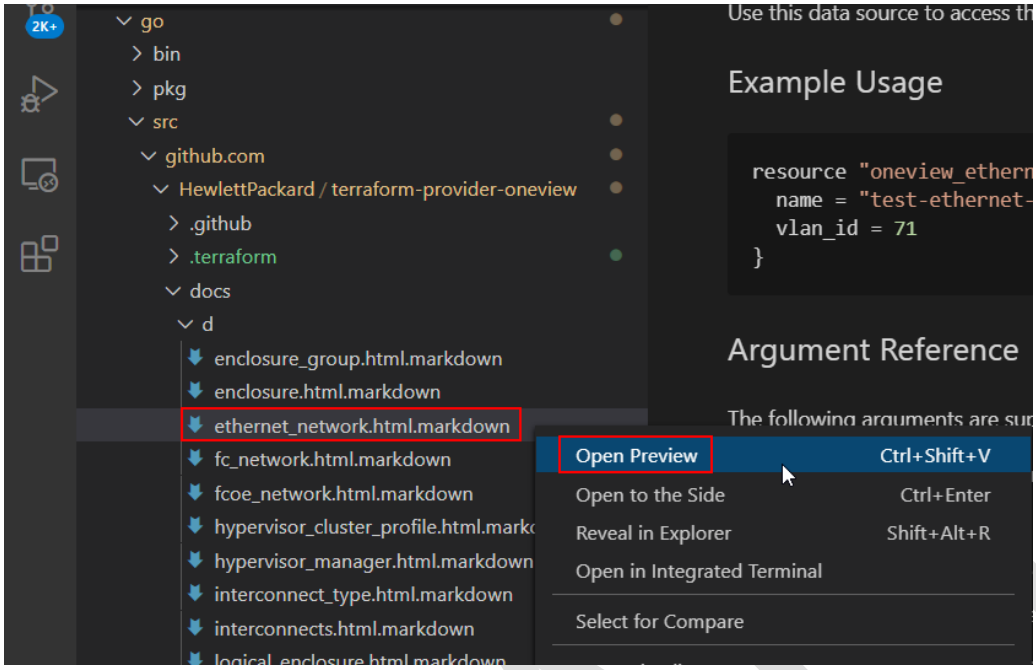
This Terraform configuration file is an example for creating and updating an ethernet network.

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

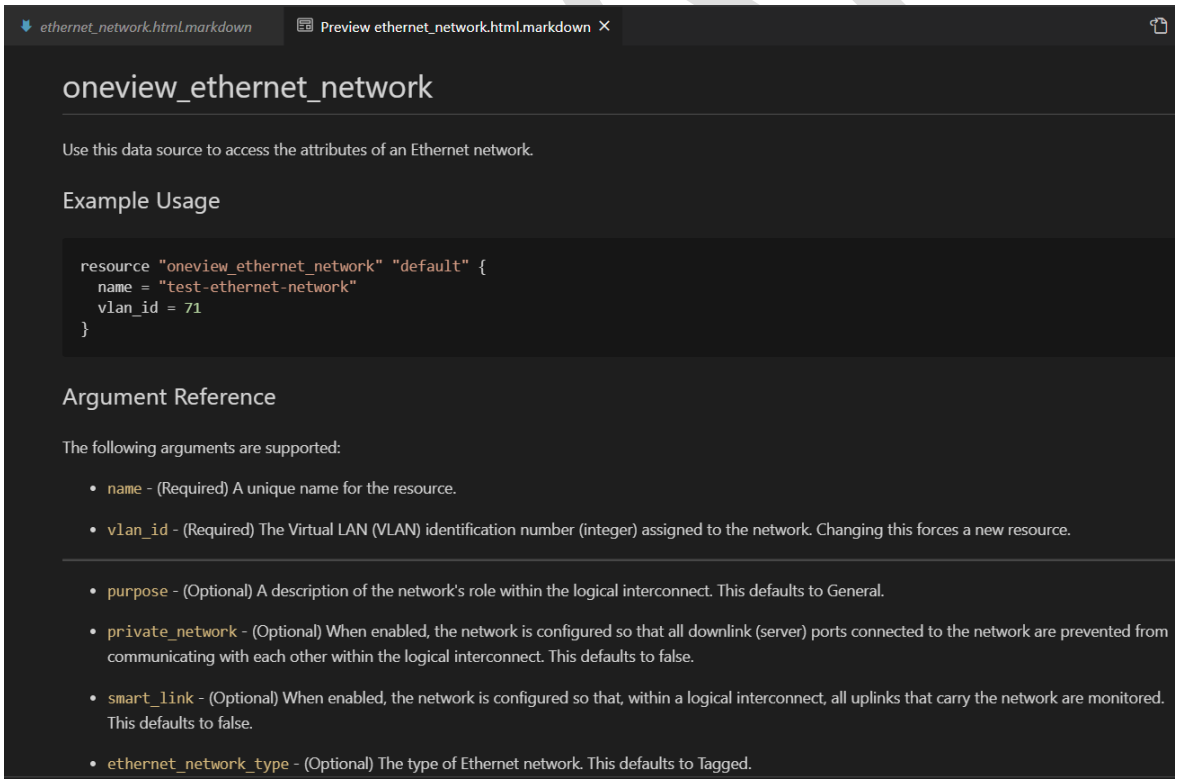
    1. ***/docs/d*** = folder documentation on how to access the attribute of a resource (**d** for data source)
    2. ***/docs/r*** = folder documentation on how to create a resource (**r** for resource)



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



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



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

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

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

```
cd ~
```

> **Important note**: It is very important to run the following commands from the Ubuntu WSL console and not from the VS Code one as you will face an illegal char error message when performing the *terraform init* command. This is due to a file encoding format produced by VS Code that Terraform is unable to read.
>
> The *terraform init* command will display the following message if run from the VC Code console:
> lionel@MIC2FZV4RN:~/terraform-create-profile $ terraform init
> There are some problems with the configuration, described below.
>
> The Terraform configuration must be valid before initialization so that
> Terraform can determine which modules and providers need to be installed.
>
> Error: Error parsing /home/*username*/terraform-create-profile/create-profile.tf: At 2:9: illegal char

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

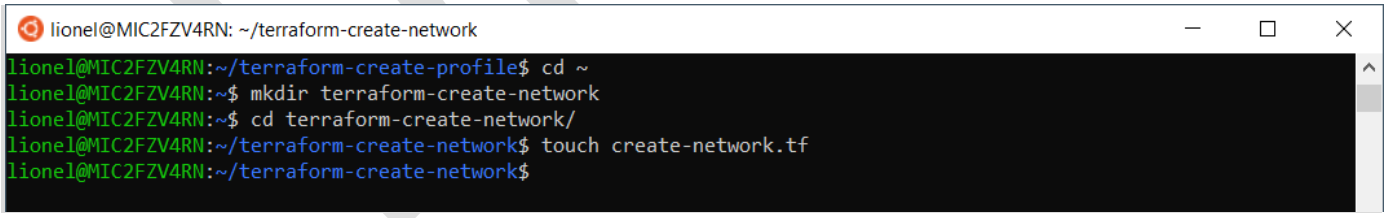
```
mkdir terraform-create-network
```

- Move into the directory:

```
cd terraform-create-network
```

- Then type:

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



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

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

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

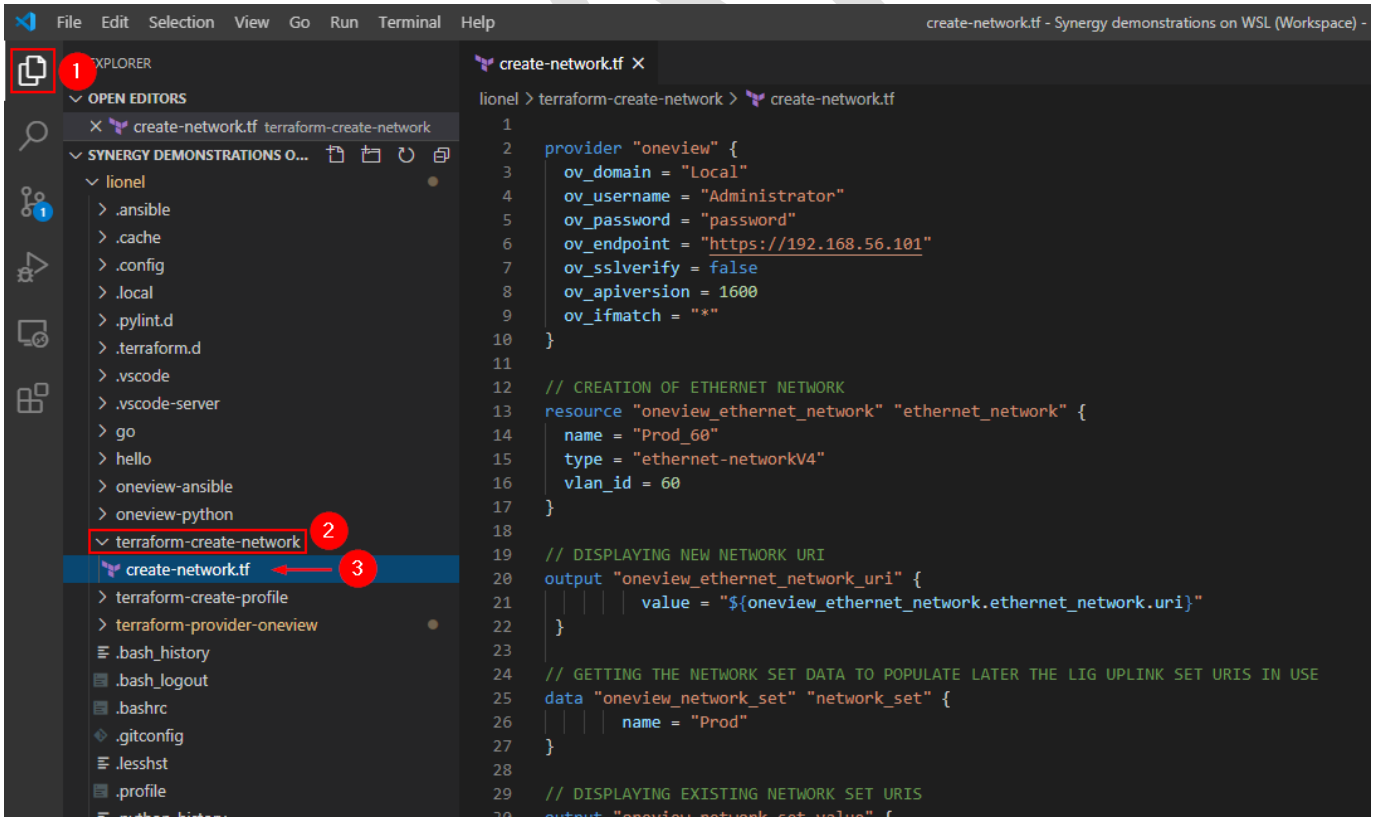
```
  ov_apiversion = 1600
  ov_ifmatch = "*"
}

// 1 - IMPORTING RESOURCE FIRST

resource "oneview_network_set" "network_set" {
}

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




// 2 - ADDING THE NEW NETWORK

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


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

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


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

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

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


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

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

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


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

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

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


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

EOF
```

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



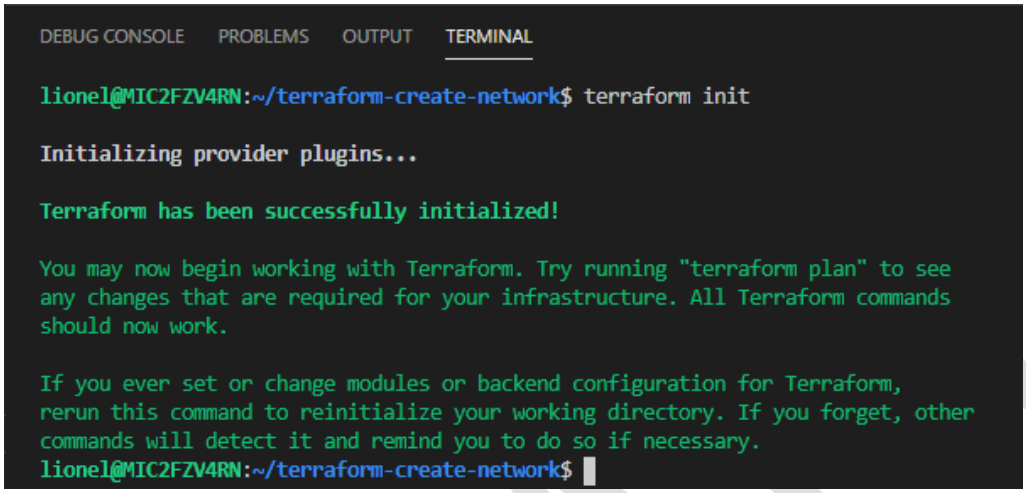
- From the VS Code console, move into the *terraform-create-network* folder:

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

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

```
terraform init
```

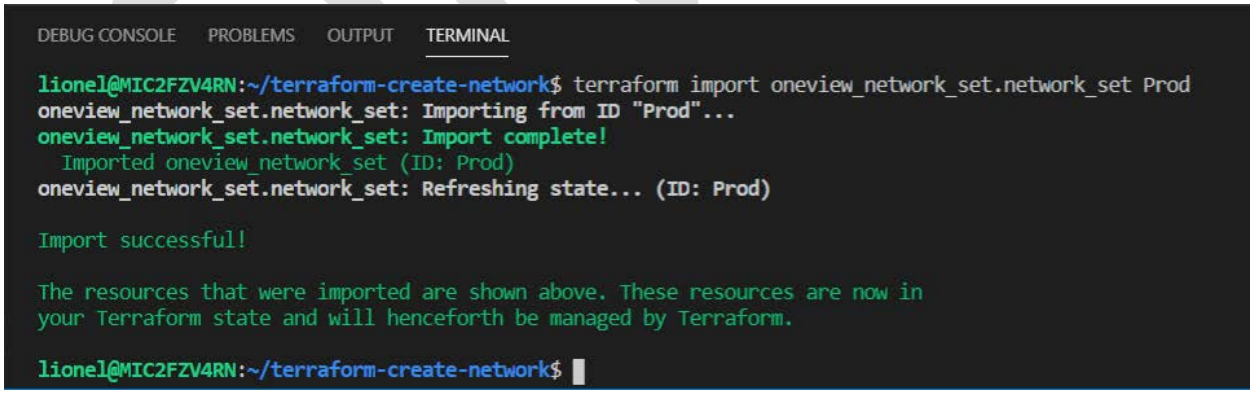
This command downloads the *oneview* provider plugin and take other actions needed to initialize our project.



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

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

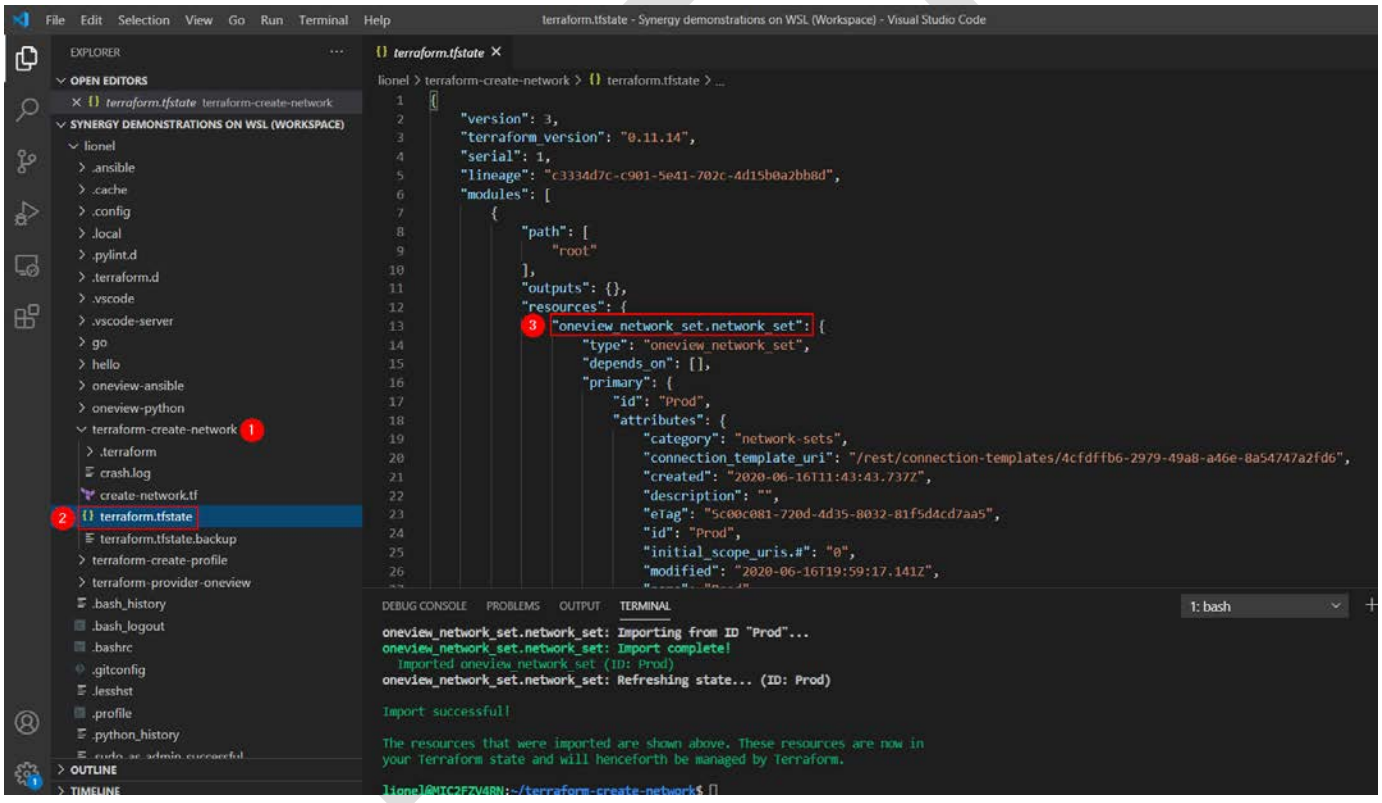
```
terraform import oneview_network_set.network_set Prod
```



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

```
// IMPORTING RESOURCE FIRST

resource "oneview_network_set" "network_set" {
}

resource "oneview_logical_interconnect_group" "logical_interconnect_group" {
}

resource "oneview_logical_interconnect" "logical_interconnect"{
}
```

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



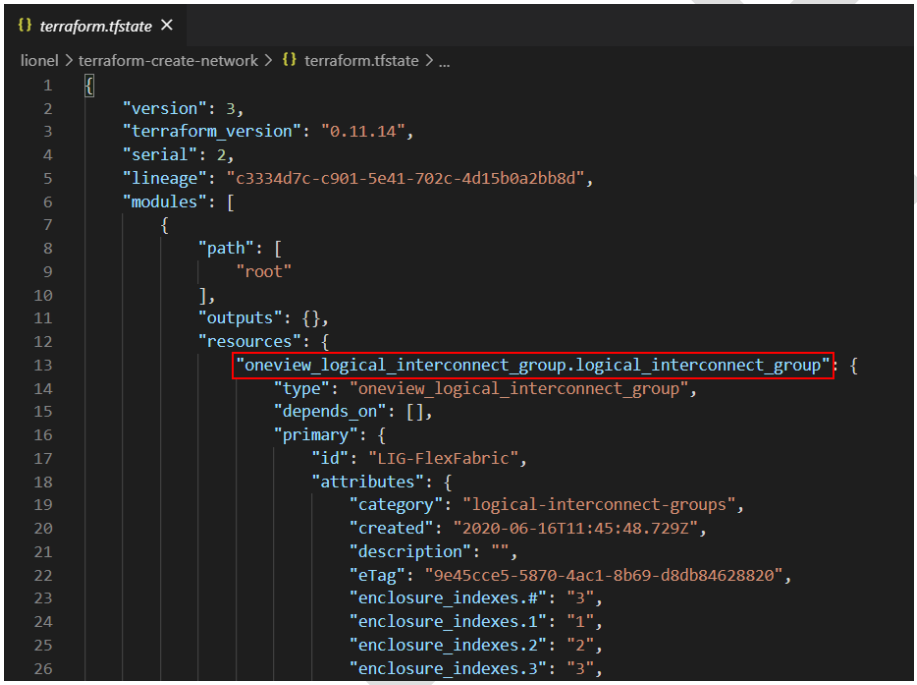
- We also need to import the LIG and LI resources. To import the LIG, we can use our FlexFabric LIG name:

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

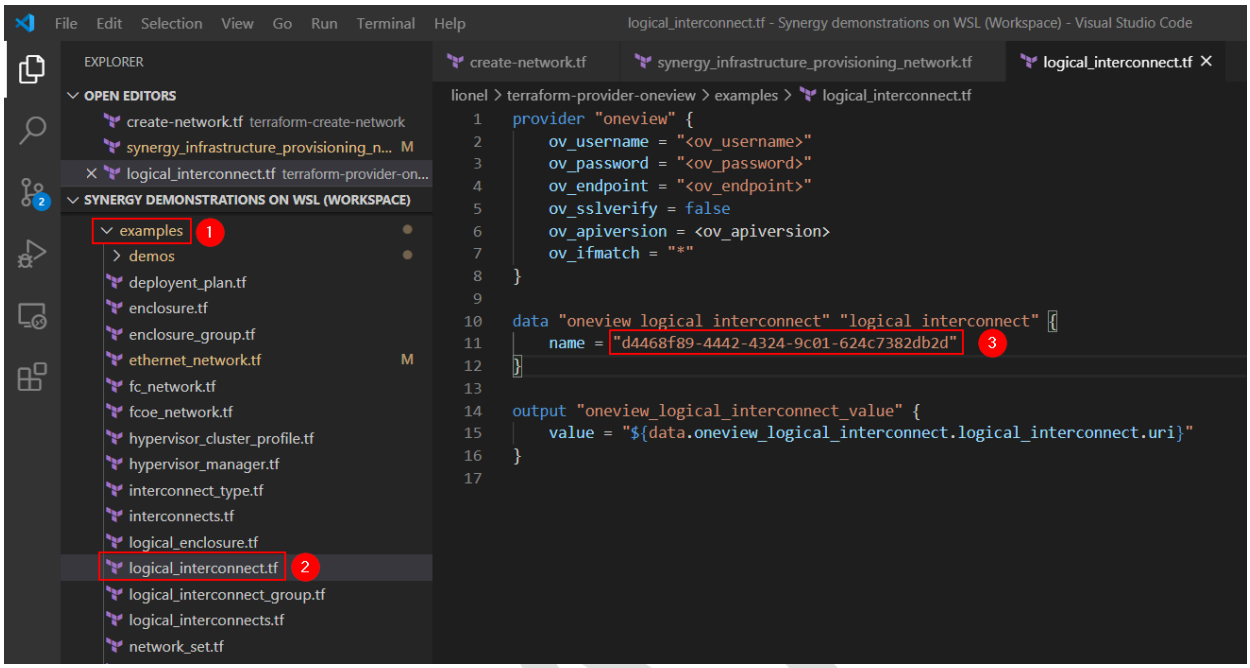
```
lionel@MIC2FZV4RN:~/terraform-create-network$ terraform import oneview_logical_interconnect_group.logical_interconnect_group LIG-FlexFabric
oneview_logical_interconnect_group.logical_interconnect_group: Importing from ID "LIG-FlexFabric"...
oneview_logical_interconnect_group.logical_interconnect_group: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

Import successful!

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

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

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

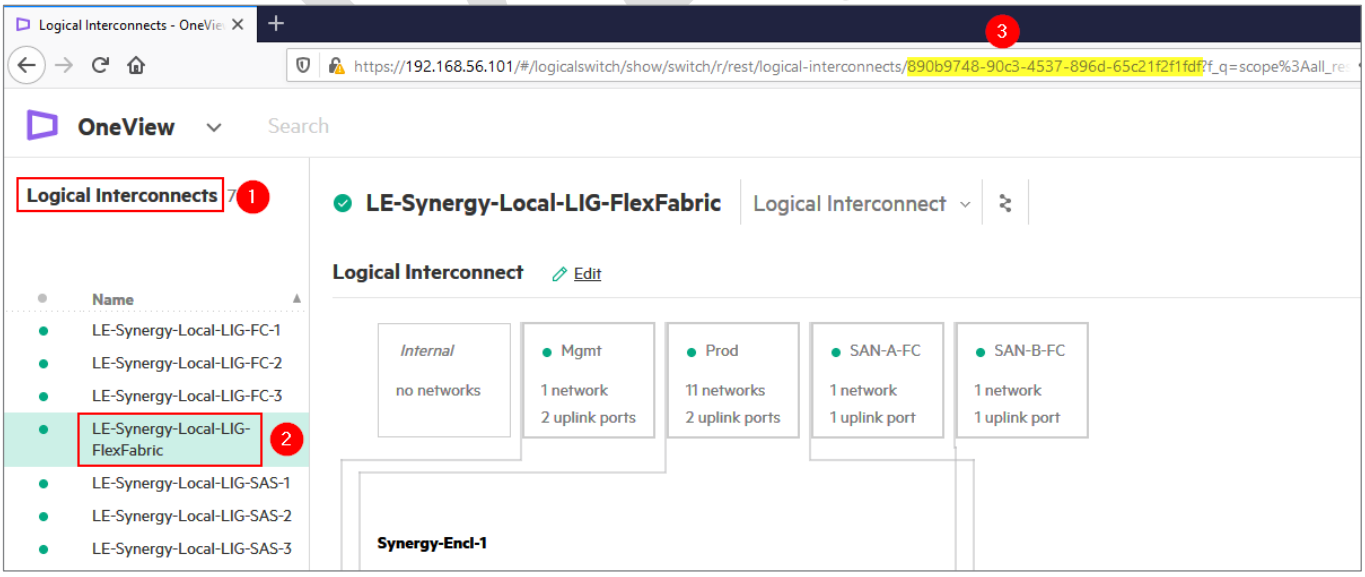
- Notice this new LIG resource with its long list of attributes has been added to the *terraform.tfstate* file:



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



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

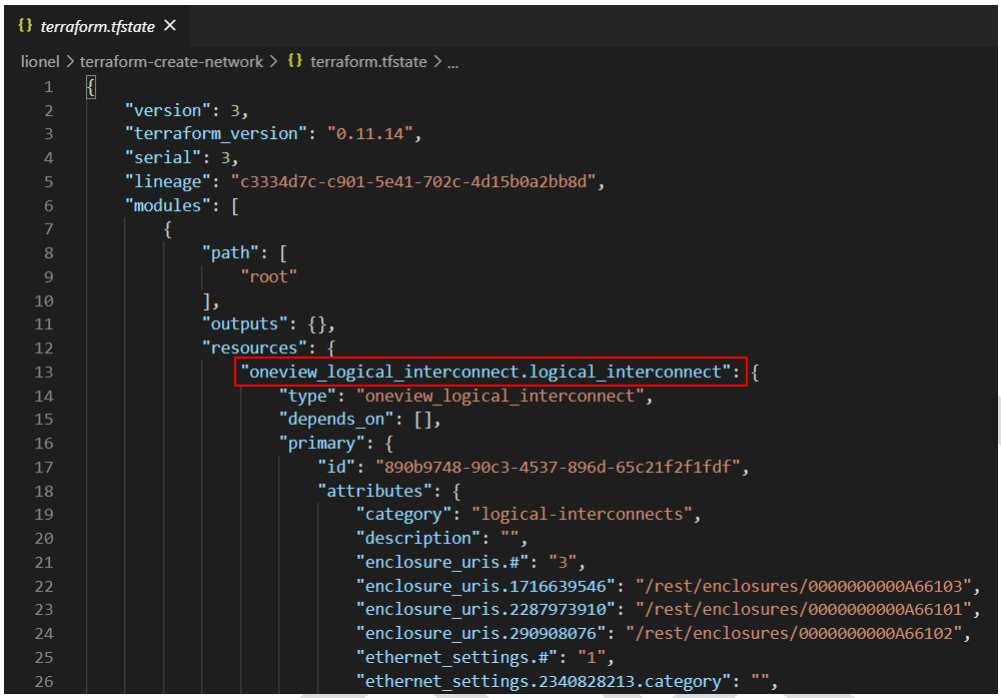


> **Note**: The ID is starting just after **/rest/logical-interconnects/** and ending before the **?**.
> For instance, the ID in this URL below is 890b9748-90c3-4537-896d-65c21f2f1fdf:
>
> how/switch/r/rest/logical-interconnects/890b9748-90c3-4537-896d-65c21f2f1fdf?f_q=scope%3Aall_resources&f_

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

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

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



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

Before the change

```
// 1 - IMPORTING RESOURCE FIRST

resource "oneview_network_set" "network_set" {
}

resource "oneview_logical_interconnect_group" "logical_interconnect_group" {
}

resource "oneview_logical_interconnect" "logical_interconnect"{
}

// 2 - ADDING THE NEW NETWORK

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

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

After the change

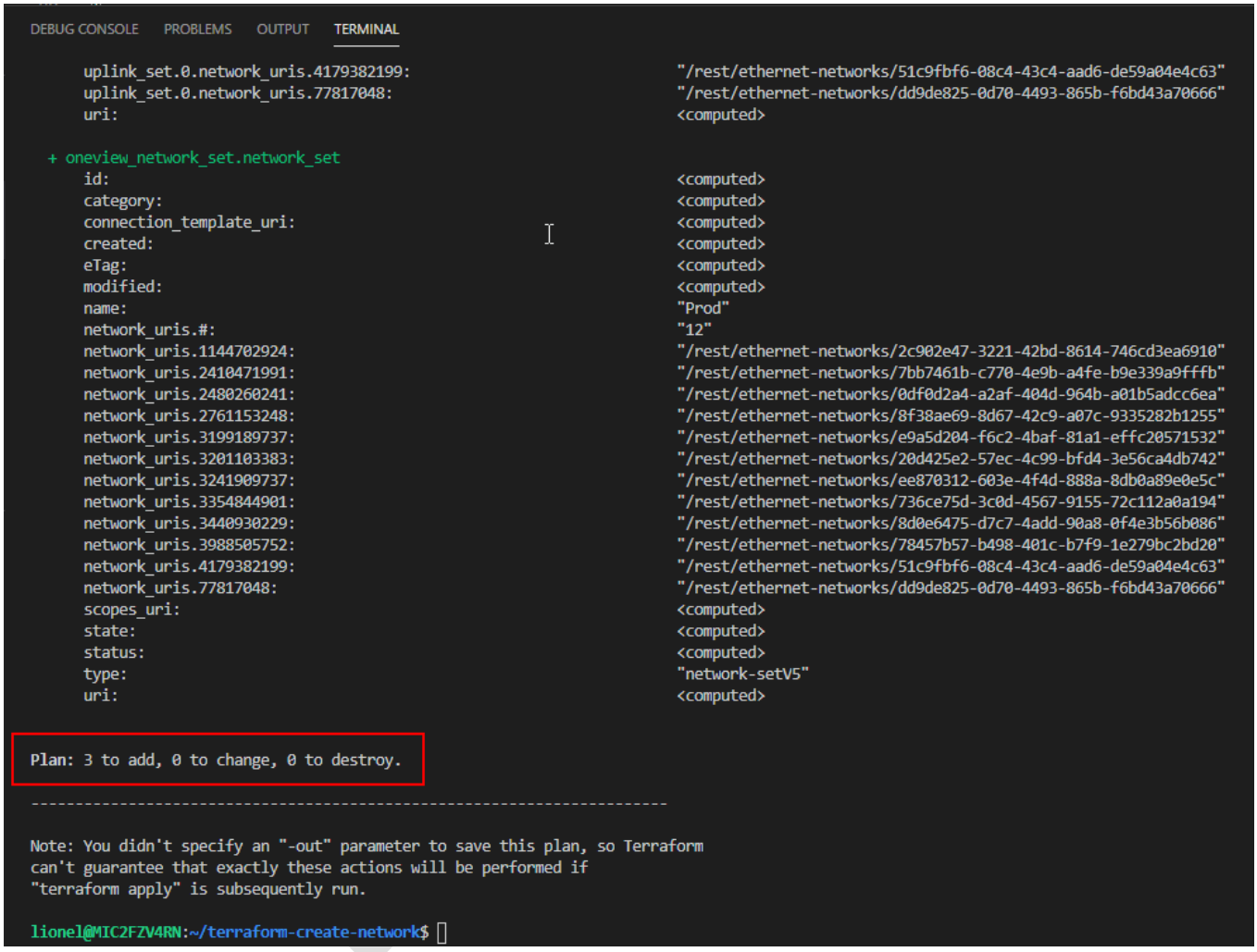
```
// 1 - IMPORTING RESOURCE FIRST

# resource "oneview_network_set" "network_set" {
# }

# resource "oneview_logical_interconnect_group" "logical_interconnect_group" {
# }

# resource "oneview_logical_interconnect" "logical_interconnect"{
# }

// 2 - ADDING THE NEW NETWORK

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

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

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

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

```
terraform plan
```

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



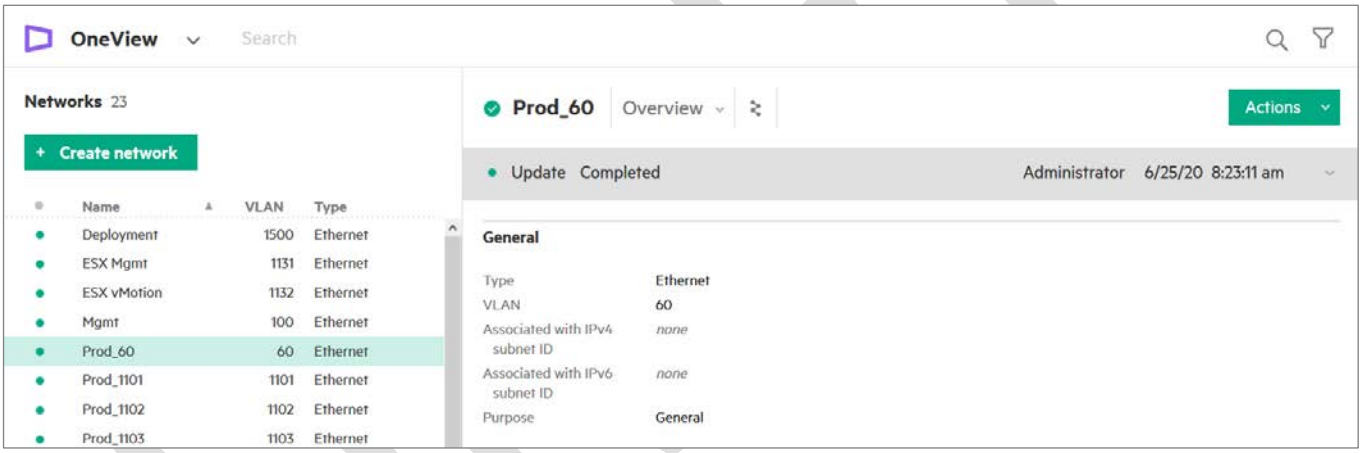
- Now that we have evaluated the plan, we can run an apply to perform all the actions:

```
terraform apply
```

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

```
      network_uris.3440930229:                    "/rest/ethernet-networks/8d0e8475-d7c7-4add-90a8-6f4e5b56b086"
      network_uris.3988505752:                    "/rest/ethernet-networks/78457b57-b498-401c-b7f9-1e279bc2bd20"
      network_uris.4179382199:                    "/rest/ethernet-networks/51c9fbf6-08c4-43c4-aad6-de59a04e4c63"
      network_uris.77817048:                      "/rest/ethernet-networks/dd9de825-0d70-4493-865b-f6bd43a70666"
      scopes_uri:                                 <computed>
      state:                                      <computed>
      status:                                     <computed>
      type:                                       "network-setV5"
      uri:                                        <computed>


Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

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



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

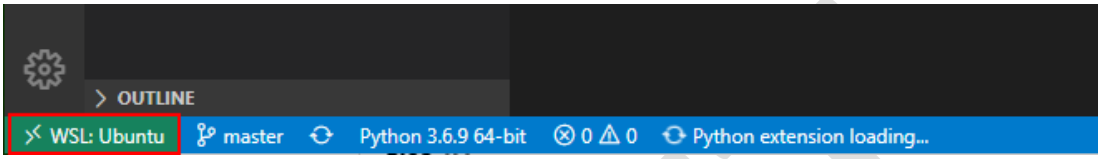
This concludes Terraform – Scenario 1

Terraform – Scenario 2 – Provisioning New Servers

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

Prerequisites:

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



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

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

- From the Ubuntu WSL console and enter:

```
cd ~
```

> **Important note**: It is very important to run the following commands from the Ubuntu WSL console and not from the VS Code one as you will face an illegal char error message when performing the *terraform init* command. This is due to a file encoding format produced by VS Code that Terraform is unable to read.
>
> lionel@MIC2FZV4RN:~/terraform-create-profile $ terraform init
> There are some problems with the configuration, described below.
>
> The Terraform configuration must be valid before initialization so that
> Terraform can determine which modules and providers need to be installed.
>
> Error: Error parsing /home/lionel/terraform-create-profile/create-profile.tf: At 2:9: illegal char

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

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

- Change to the directory:

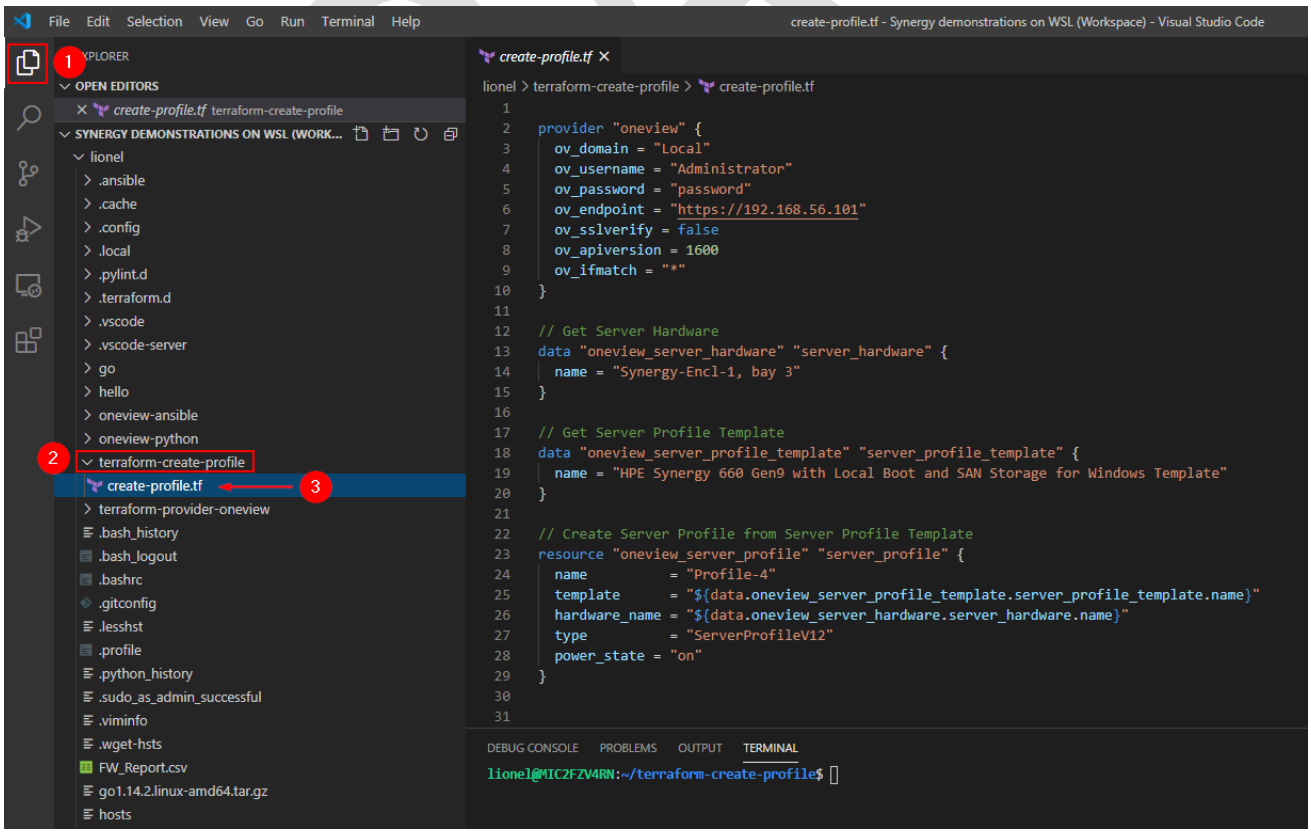
```
cd terraform-create-profile
```

- Then create the following Terraform configuration file using:

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

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

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

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

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

EOF
```

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

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

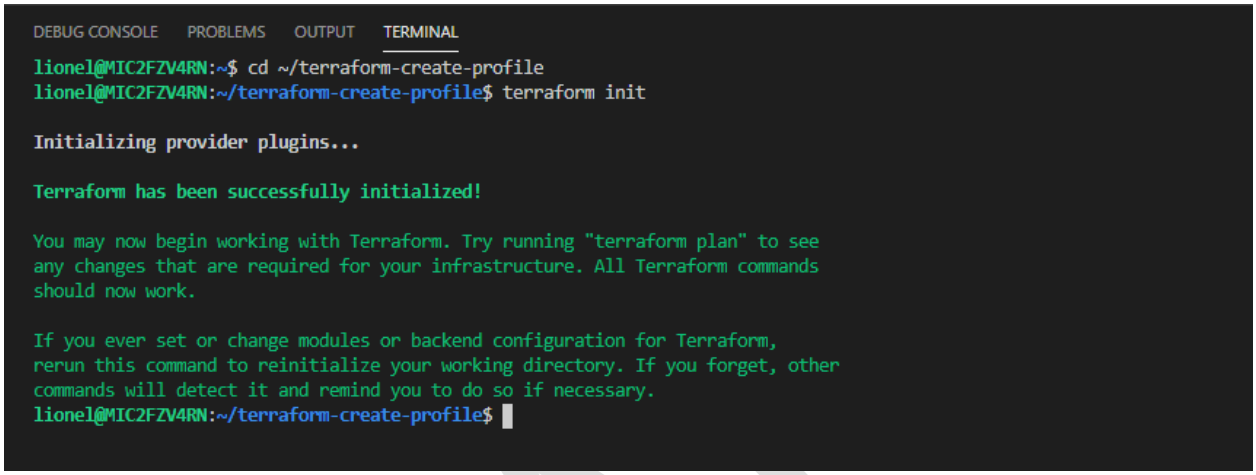


- From the VS Code console, go to the *terraform-create-profile* folder:

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

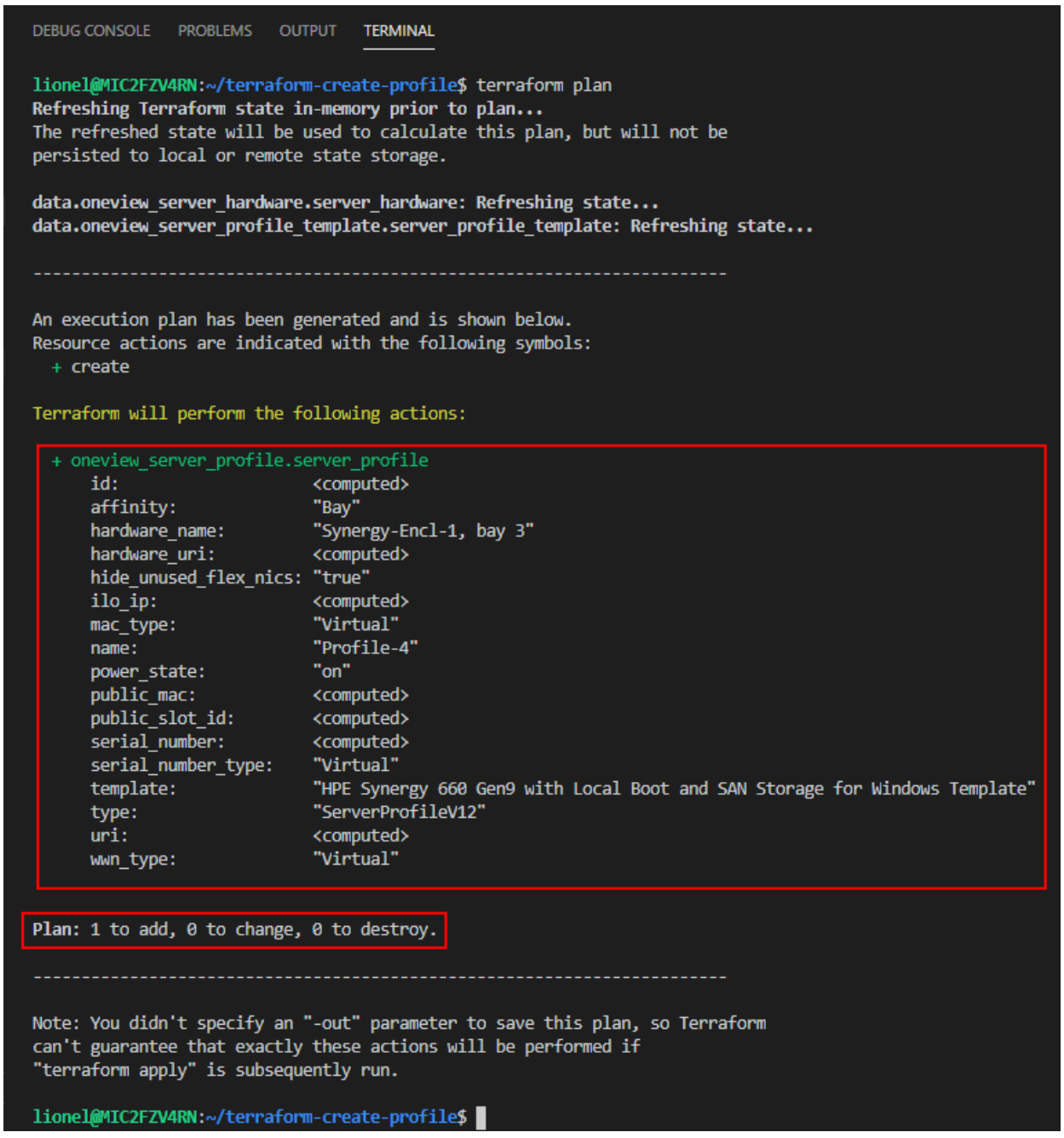
- Now we need to run the *init* command which will prepare Terraform in this directory:

```
terraform init
```



- Next, we can verify that Terraform will create the server profile resource as we expect before making any actual changes to our infrastructure, enter:

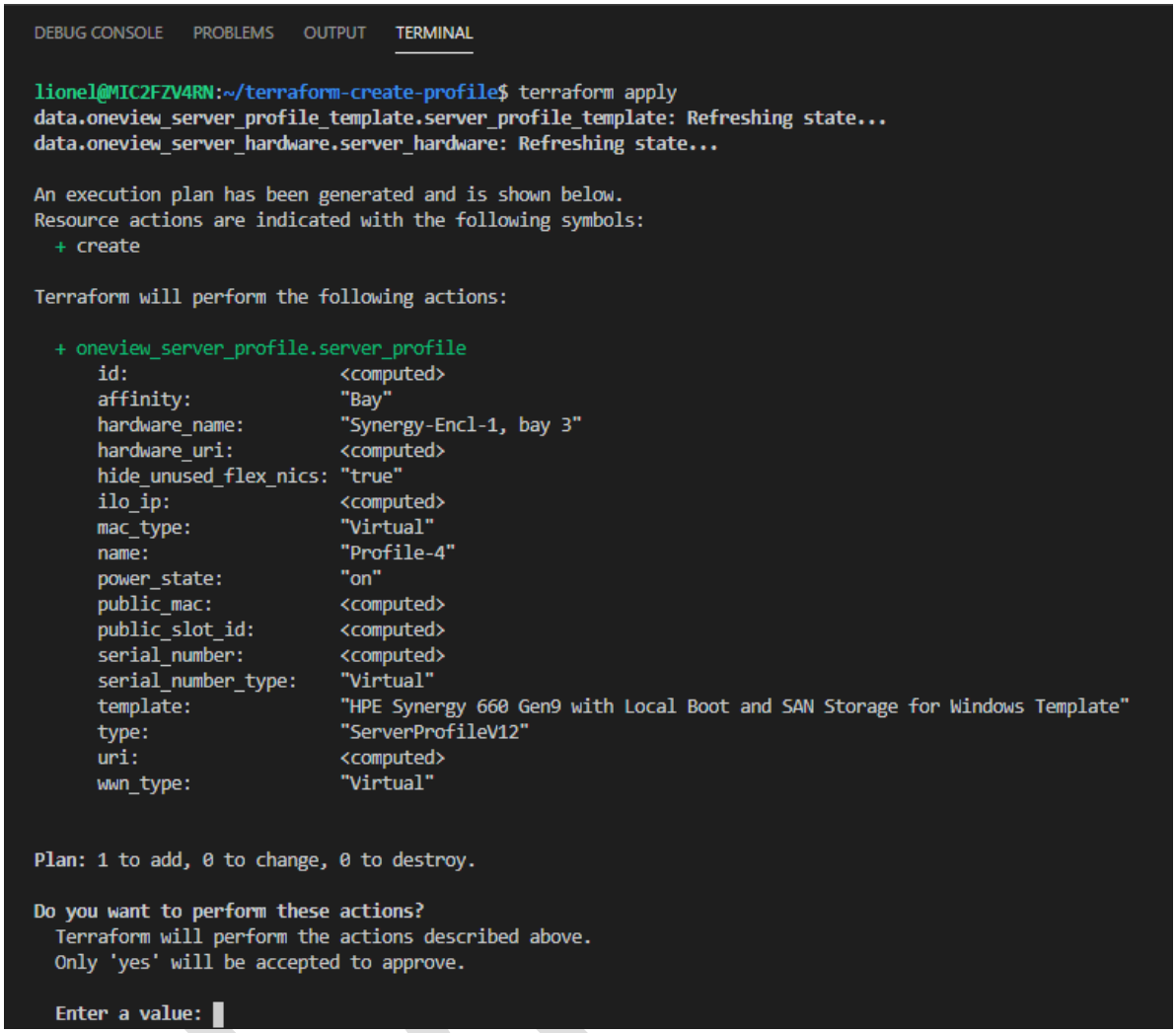
```
terraform plan
```



As we can see the only pending action is the creation of the profile.

- Now that we have evaluated the plan, we can run an apply:

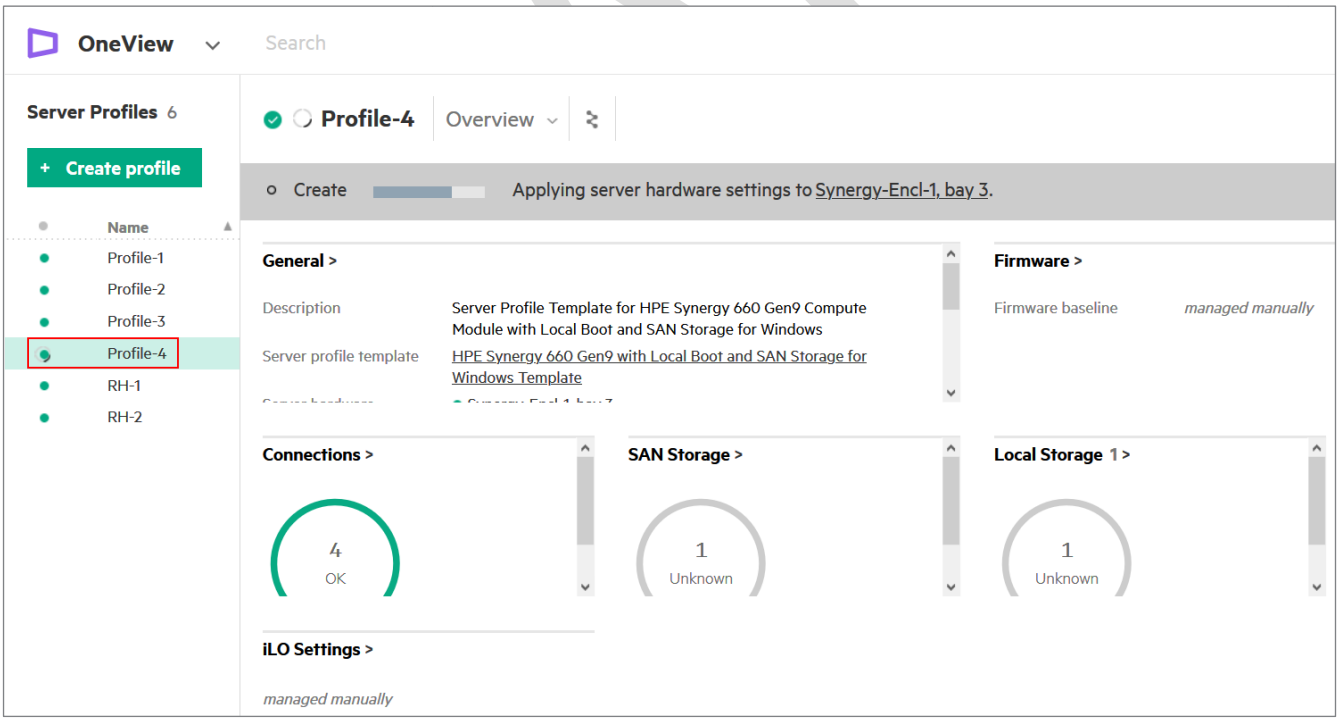
```
terraform apply
```



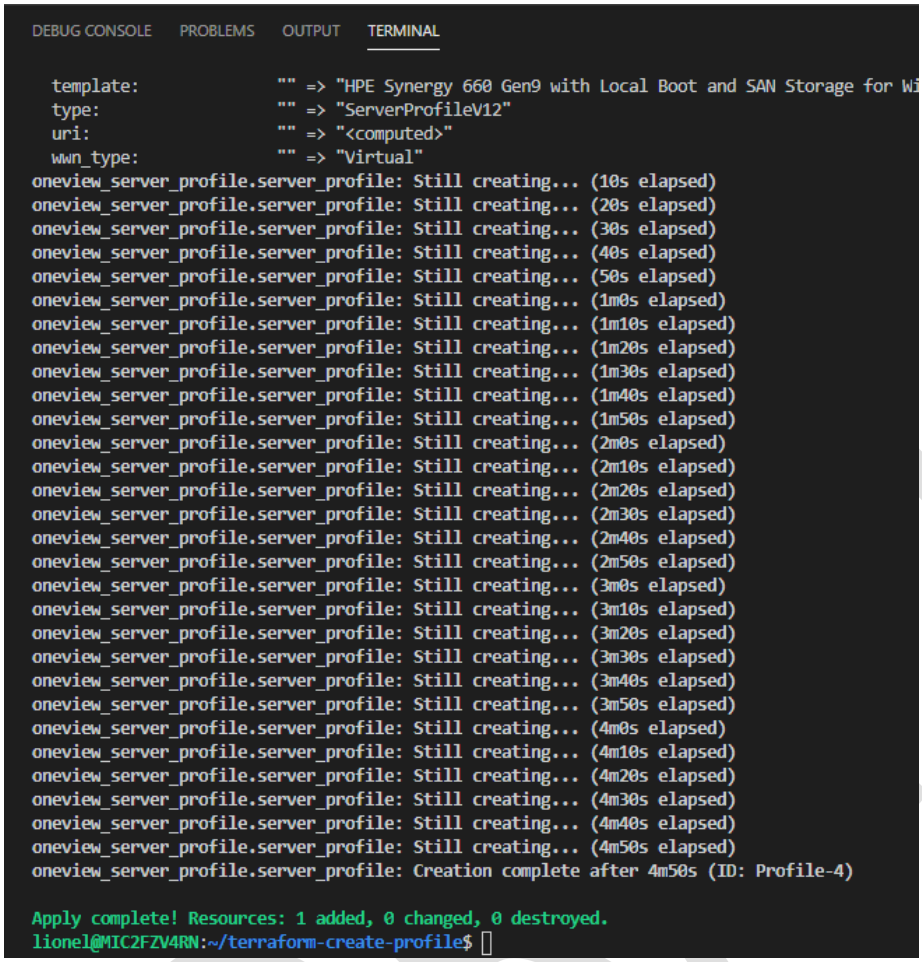
- When prompted, type **yes** to confirm that you want to perform the server profile creation.

```
  Enter a value: yes

oneview_server_profile.server_profile: Creating...
  affinity:              "" => "Bay"
  hardware_name:         "" => "Synergy-Encl-1, bay 3"
  hardware_uri:          "" => "<computed>"
  hide_unused_flex_nics: "" => "true"
  ilo_ip:                "" => "<computed>"
  mac_type:              "" => "Virtual"
  name:                  "" => "Profile-4"
  power_state:           "" => "on"
  public_mac:            "" => "<computed>"
  public_slot_id:        "" => "<computed>"
  serial_number:         "" => "<computed>"
  serial_number_type:    "" => "Virtual"
  template:              "" => "HPE Synergy 660 Gen9 with Local Boot and SAN Storage for Windows Template"
  type:                  "" => "ServerProfileV12"
  uri:                   "" => "<computed>"
  wwn_type:              "" => "Virtual"
oneview_server_profile.server_profile: Still creating... (10s elapsed)
oneview_server_profile.server_profile: Still creating... (20s elapsed)
oneview_server_profile.server_profile: Still creating... (30s elapsed)
oneview_server_profile.server_profile: Still creating... (40s elapsed)
oneview_server_profile.server_profile: Still creating... (50s elapsed)
```

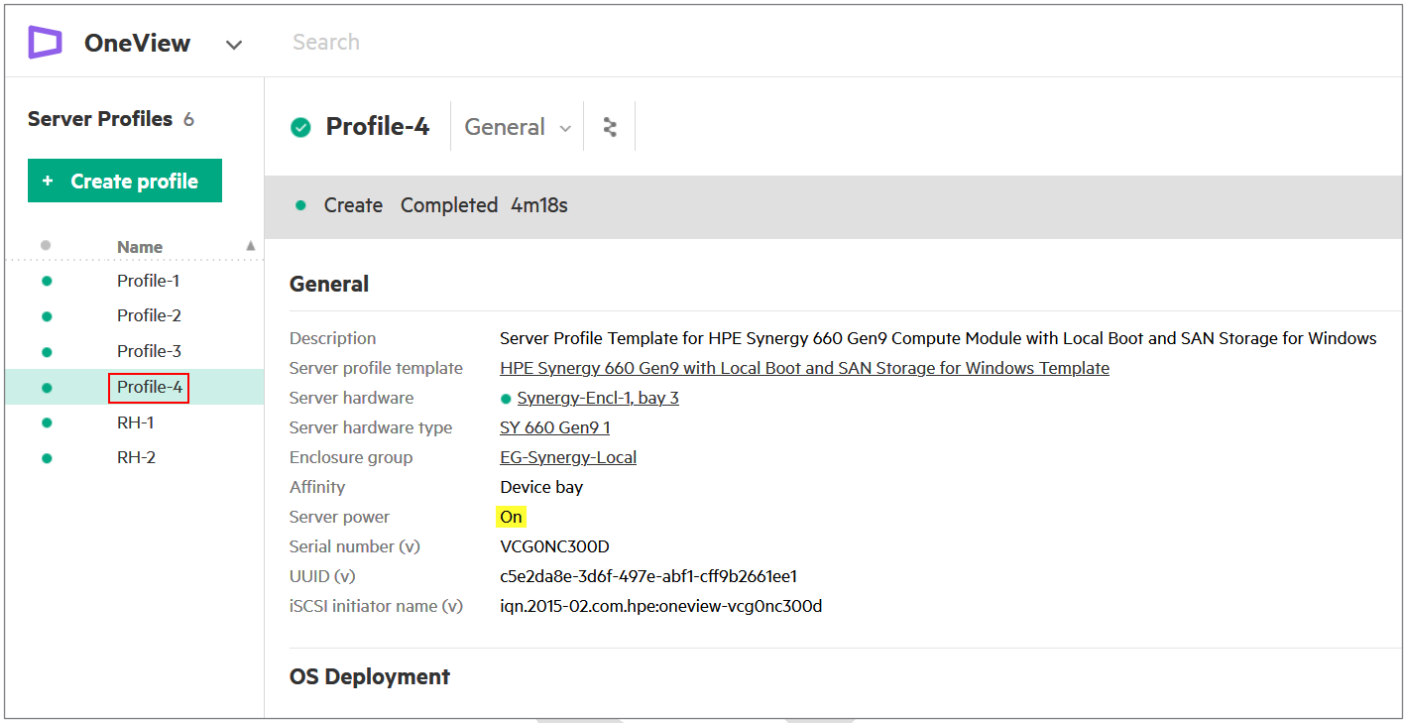
- Open the OneView UI (https://192.168.56.101/#/profiles) to show the creation of the new server profile:



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



- Once completed, a new Server Profile *Profile-4* is created, and the server is powered-on as requested in the Terraform configuration file using `power_state = "on"`



This concludes Terraform – Scenario 2

Terraform – Scenario 3 – Unprovisioning Running Server

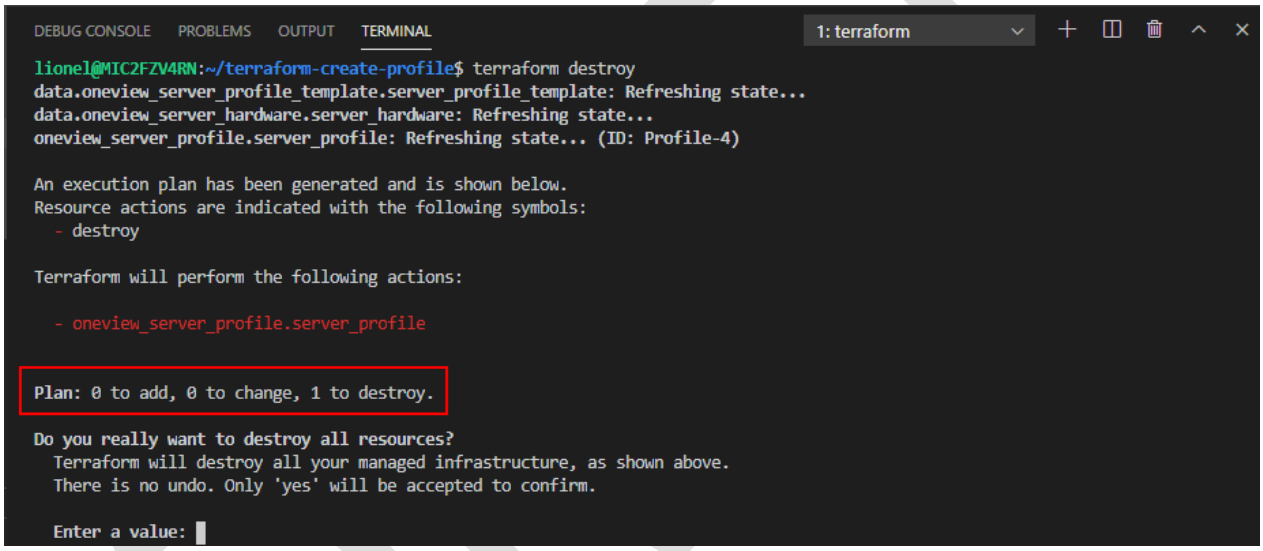
Now that the server that has been created in scenario 2 is up and running, we can explain to the customer that as easily as the provisioning has been made, we can now unprovision the server and return the server hardware back to the OneView resource pool.

> **Note**: This scenario can only be run after *Terraform – Scenario 2* when server profile *Profile-4* is available in OneView.

To do this using Terraform is incredibly easy as there is no need to create a new configuration file but just to destroy this configuration project using the destroy command.

- From the VS Code console, enter:

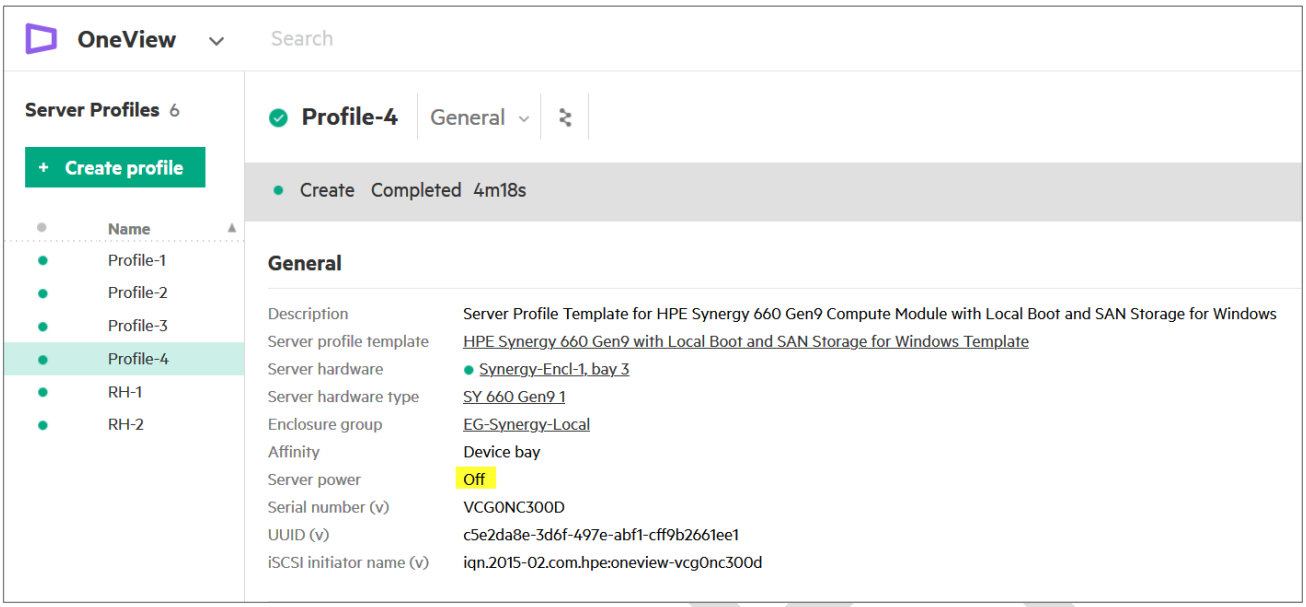
```
terraform destroy
```



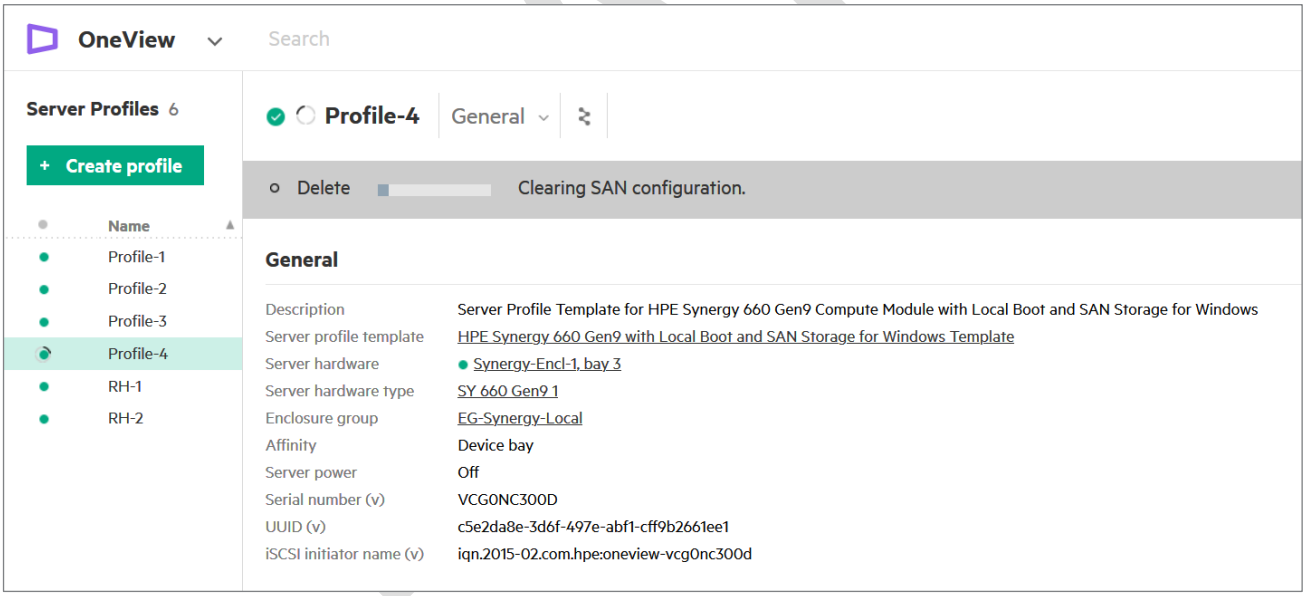
  This command generates a report detailing what actions Terraform will take to destroy our oneview resources. We can see that Terraform will perform 1 action to destroy the server profile resource.

- Enter **Yes** to proceed the destroy action.

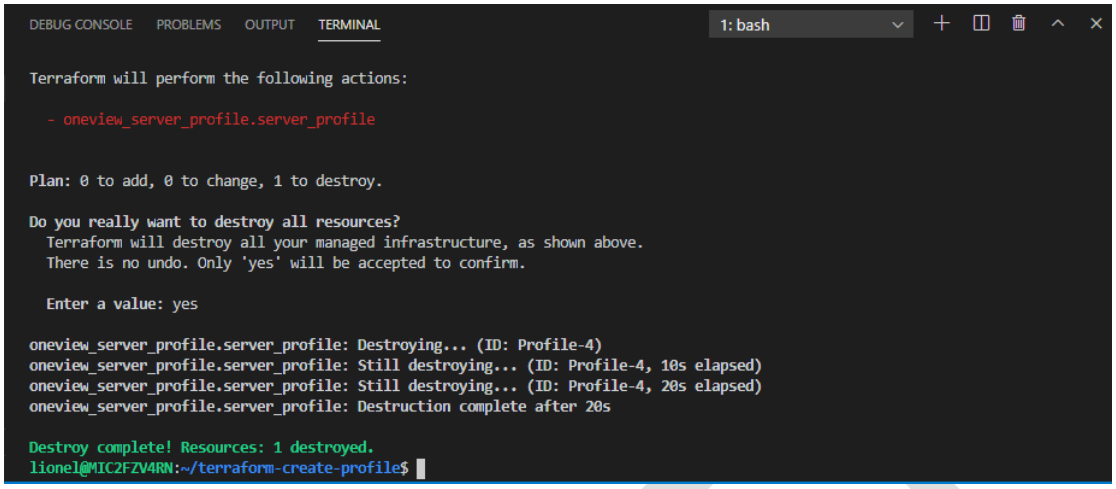
- In the first place, the server is powered off:



- Then the profile is deleted:



- In the terminal, the destroy is complete message is displayed with 1 resource destroyed:



This concludes Terraform – Scenario 3

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

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

Chapter-8 - How to reset the Synergy demonstration environment

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

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

1- First snapshot: OneView appliance first time setup is done (IP addresses set, discovery of all enclosures and servers is done) but the Synergy frames are not configured (no LE, no LIG, no EG, no network)

   ⇨ Can be used to show how to automate the setup of Synergy and the power of our infrastructure as code implementation.

2- Second snapshot: OneView appliance is fully configured with LE, EG, LIG and some networks.

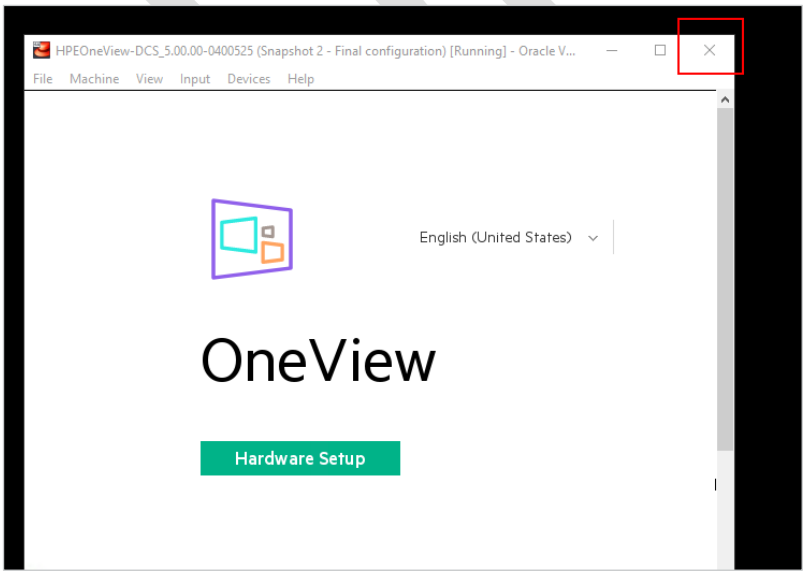
   ⇨ Can be used to run demos with an already configured environment to demonstrate features of the Composable infrastructure like creating server profiles, adding networks, modifying VC configuration, etc.

Shutting down the Synergy Composer Demonstration appliance

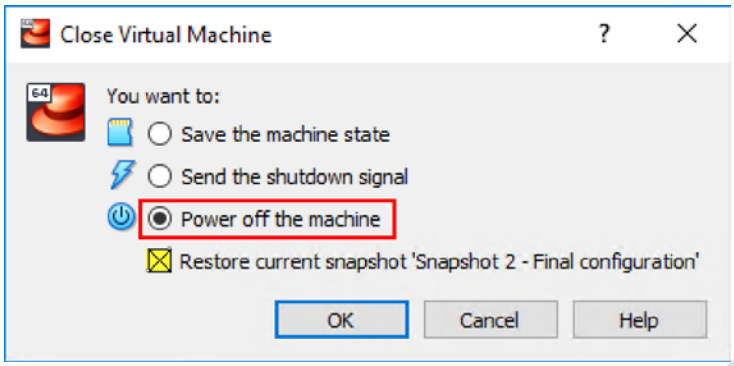
With VirtualBox, when shutting down a virtual machine, you have a "Restore current snapshot" option if you want VirtualBox to load the most recent snapshot the next time you start up the virtual machine again. This is a very convenient and easy option to restore the appliance back to the pre-configured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

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

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



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



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

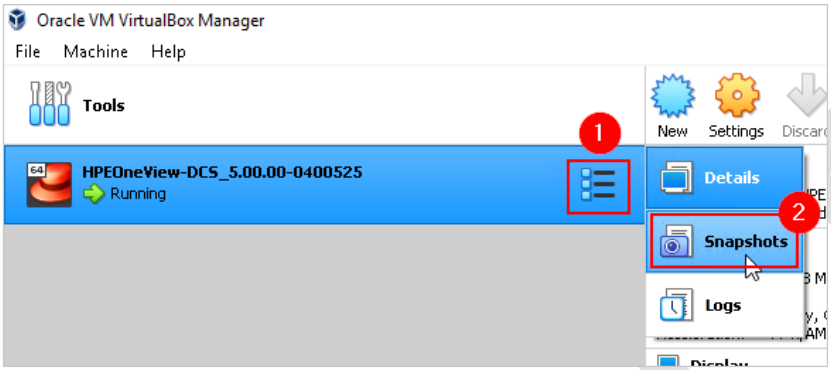
> **Note**: By default, VirtualBox shutdown dialog provides a restore snapshot option with the latest created snapshot.

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

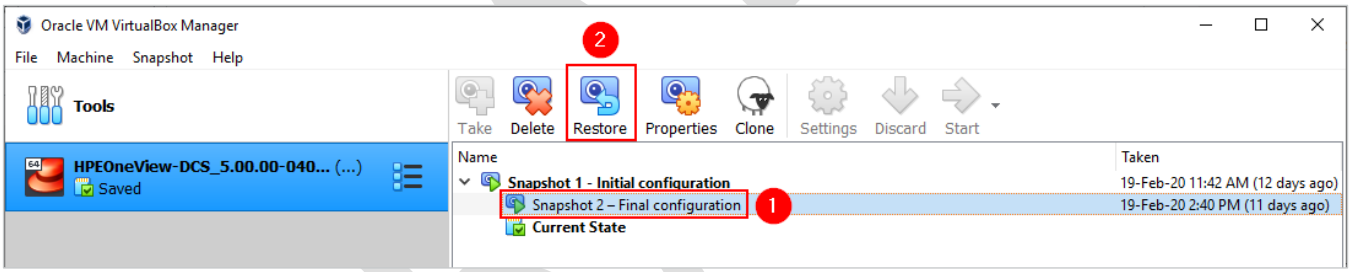
Resetting the Synergy Composer Demonstration appliance to the fully configured state

To start the appliance with the Synergy frames fully configured (with LE, LIG, EG and networks) in order to run the demos and demonstrate features of the Composable infrastructure, we need to restore the second snapshot.

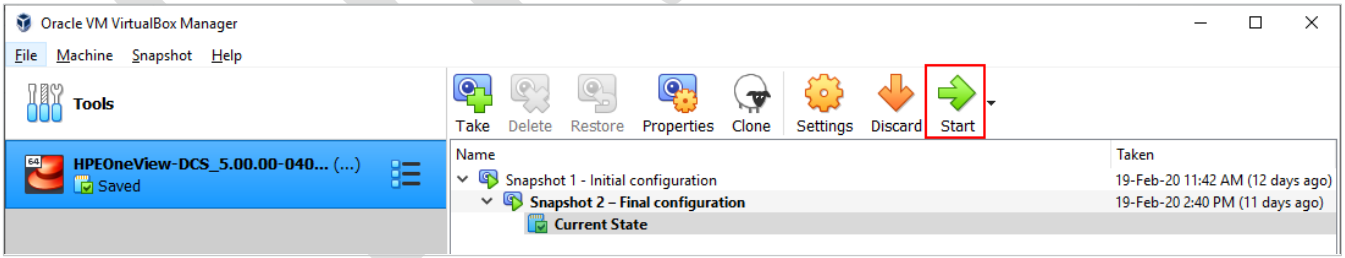
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- Select **Snapshot 2 – Final configuration** then click **Restore**



- Then start the appliance by selecting **Start**

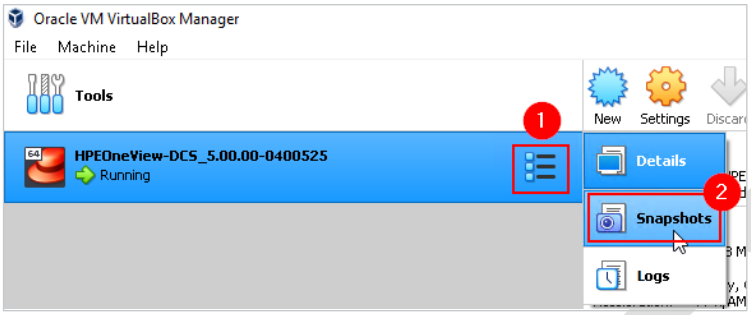


> **Note**: With VirtualBox, rolling back to one of the previous snapshots can only be done when a VM is not running.
>
> **Note**: When you start the DCS appliance using the snapshots prepared in this document, you should never see OneView starting and taking several minutes to start. If you see OneView starting, shutdown the VM and start again using the earlier procedure.

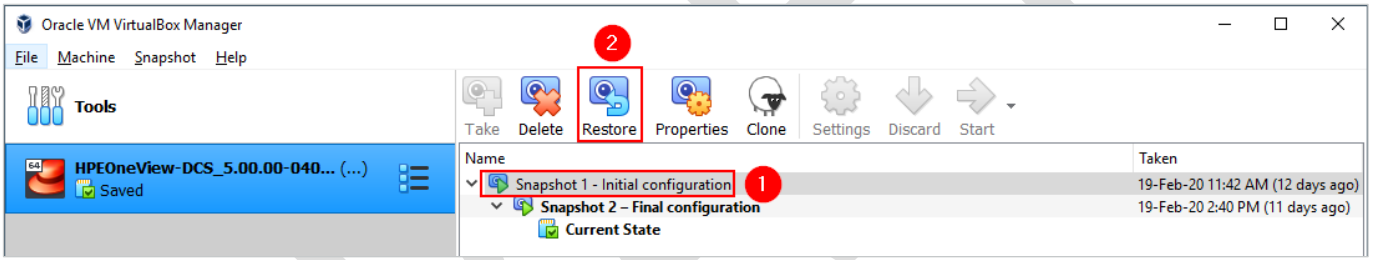
Resetting the Synergy Composer Demonstration appliance to the unconfigured state

To start the appliance with the Synergy frames unconfigured (no LE, no LIG, no EG, no network) in order to show how to automate the setup of Synergy, we need to restore the first snapshot.

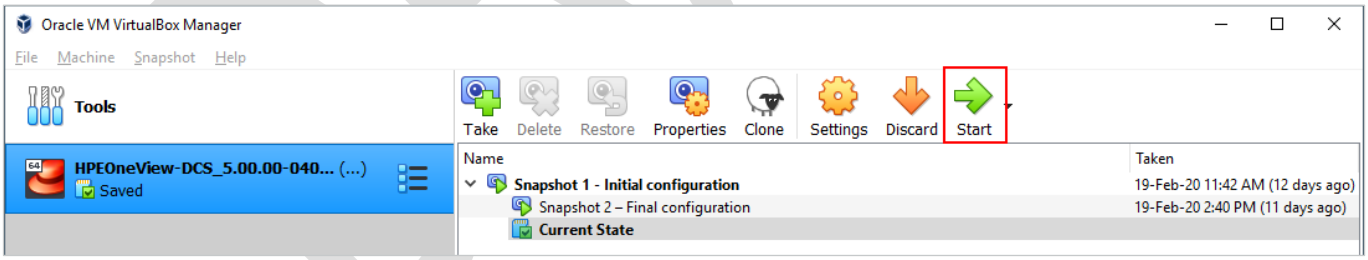
- Select the **Snapshots** option from the **Tools** menu of the DCS appliance:



- Select **Snapshot 1 – Initial configuration** then click **Restore**



- Then start the appliance by selecting **Start**



> **Note**: With VirtualBox, rolling back to one of the previous snapshots can only be done when a VM is not running.
>
> **Note**: When you start the DCS appliance using the snapshots prepared in this document, you should never see OneView starting and taking several minutes to start. If you see OneView starting, shutdown the VM and start again using the earlier procedure.

Chapter-9 - How to upgrade the Synergy demonstration environment

This chapter describes how you can upgrade the different components of your Synergy demonstration environment.

Upgrading the HPE OneView demonstration appliance (DCS)

It is not supported to upgrade the DCS appliance using the standard OneView upgrade procedure, you need to download a new version of the HPE OneView demonstration appliance and re-install it from scratch using the same steps provided in this guide.

Upgrading Ansible modules for HPE OneView

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

```
cd $home
rm -r -f oneview-ansible
git clone https://github.com/HewlettPackard/oneview-ansible.git
cd oneview-ansible
pip3 install -r requirements.txt
```

> **Note**: make sure you also merge your branch with the *bug_fix/create_profiles_in_parallel* after the upgrade. See Installing Ansible Modules for HPE OneView on Ubuntu WSL

Upgrading the Terraform provider for HPE OneView

If an old version of the Terraform provider for HPE OneView is already installed on your system, you can use the following commands to update to the latest version:

```
cd /usr/local/bin
go get -u github.com/HewlettPackard/terraform-provider-oneview
sudo mv $GOPATH/bin/terraform-provider-oneview .
```

To update the Terraform provider for HPE OneView GitHub repository clone from your home directory, enter:

```
cd ~
rm -rdf terraform-provider-oneview
sudo git clone https://github.com/HewlettPackard/terraform-provider-oneview.git
```

Upgrading HPE OneView Python SDK

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

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

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

Upgrading the PowerShell library for OneView

If a previous major version of the module is already installed on your system, you can use the following commands to uninstall the old version and install the latest version:

```
Get-Module HPOneView.500 -ListAvailable | Uninstall-Module
Install-Module -Name HPOneView.520
```

If you are not running the latest major version of the HPOneView module, simply enter:

```
Update-Module HPOneView.520 -Force
```

Troubleshooting VirtualBox VM not starting

Problem:

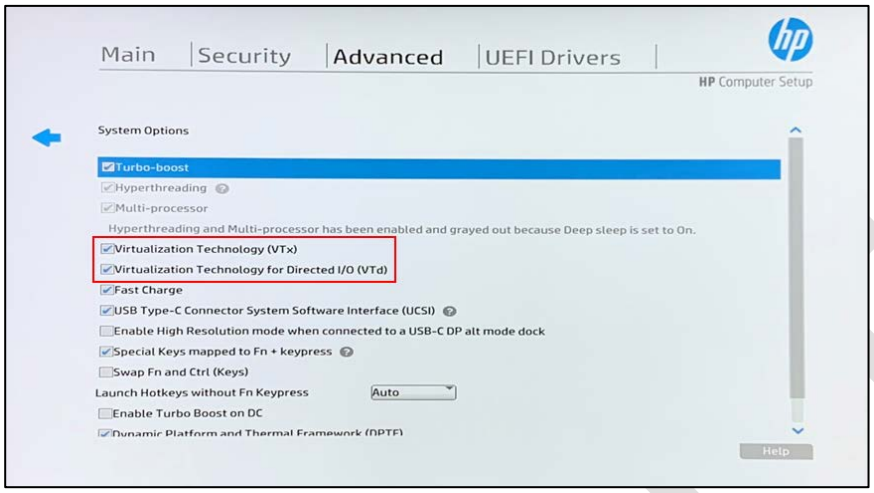
You are seeing a **VT-x is not available (VERR_VMX_NO_VMX)** in VirtualBox when starting the VM

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

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

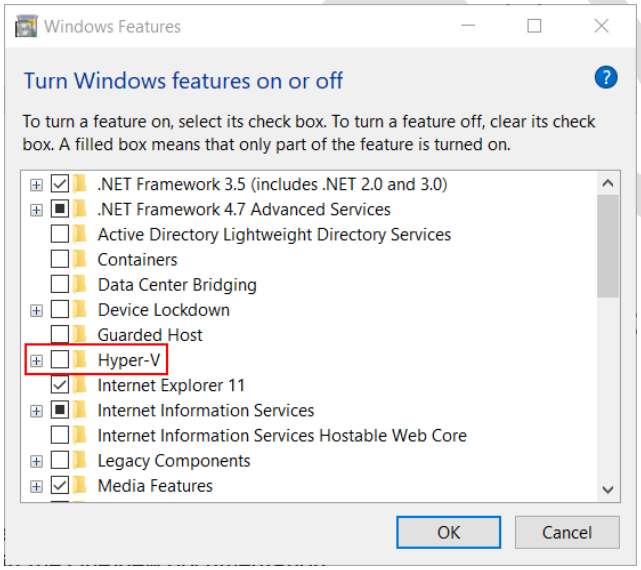
Solution:

- Make sure Virtualization (VT-x) is enabled in your BIOS.



> **Note**: On HP EliteBook 840 you must repeatedly press the **Esc** key until the Startup Menu opens. Press **F10** to enter the BIOS Setup Utility. Go to **Advanced** / **System Options**

- Make sure *Hyper-V* is disabled from Windows features. (Run | OptionalFeatures.exe)

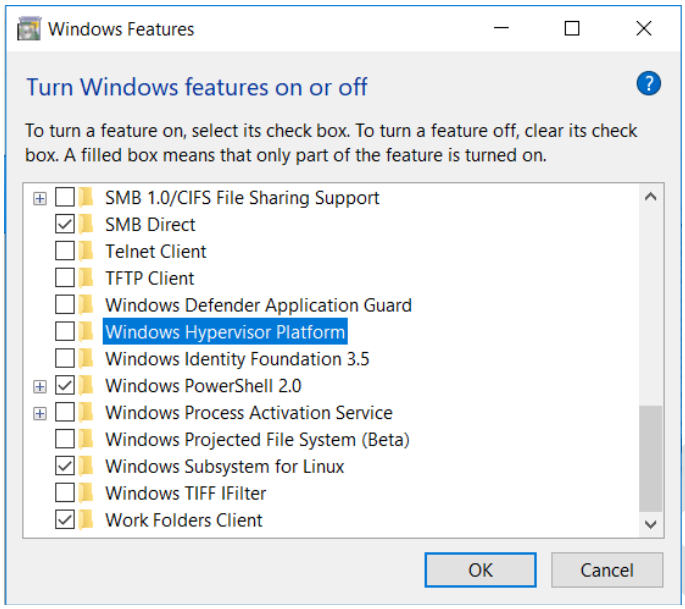


You can also run the following command to disable Hyper-V:

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

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

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

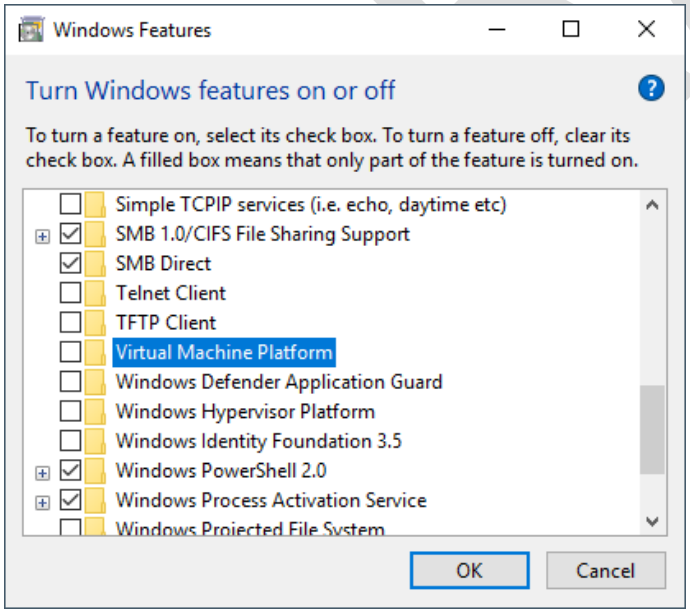


Or run:

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

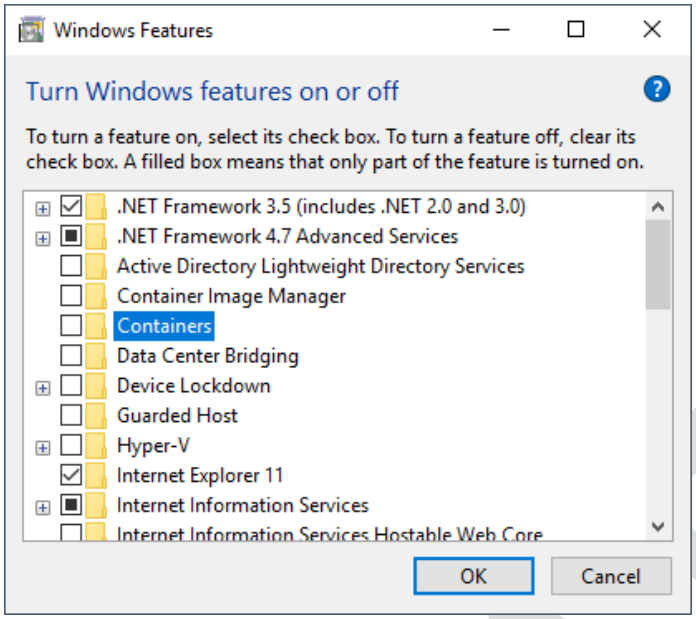
> **Note**: If running, *Windows Hypervisor Platform* is taking exclusive use of VT-x so it is required to turn it off

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



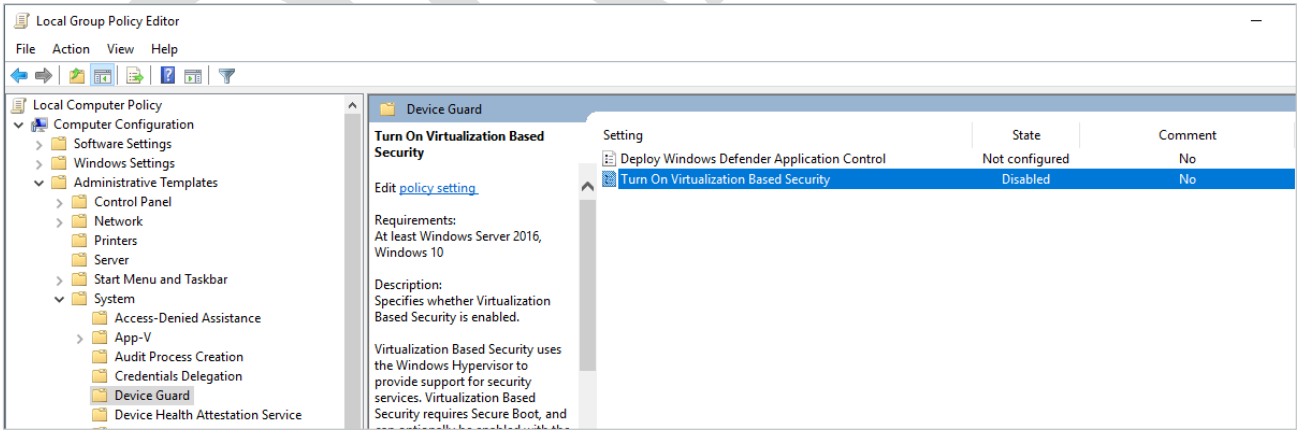
> **Note**: If running, *Virtual Machine Platform* is taking exclusive use of VT-x so it is required to turn it off

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



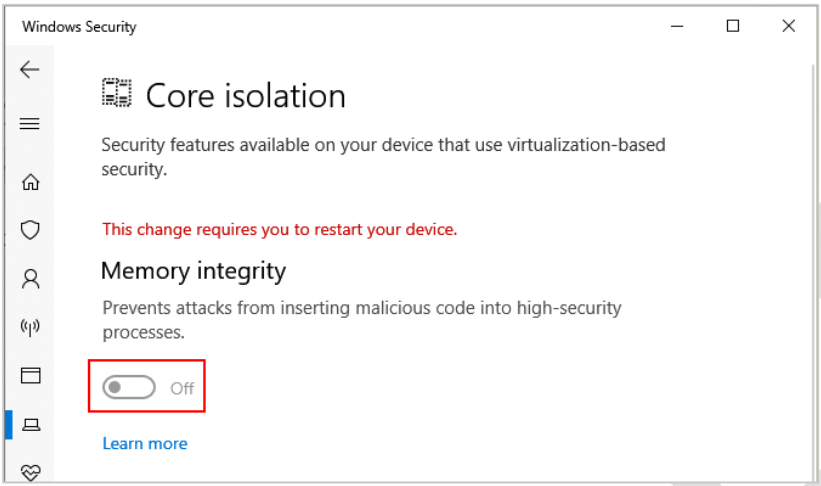
> **Note**: If running, *Container* is taking exclusive use of VT-x so it is required to turn it off

- On some Windows hosts with an EFI BIOS, Device Guard or Credential Guard may be active by default and interferes with OS level virtualization apps in the same way that Hyper-V does. These features need to be disabled. On Pro versions of Windows, you can do this using **gpedit.msc** then set **Local Computer Policy** > **Computer Configuration** > **Administrative Templates** > **System** > **Device Guard** > Turn **Virtualization Based Security** to **Disabled**.



If you cannot use gpedit for some reason, then the equivalent registry hack is to find the key HKLM|SYSTEM|CurrentControlSet|Control|DeviceGuard|EnableVirtualizationBasedSecurity|Enabledand set it to 0.

- On Win10 hosts, check **Windows Defender** > **Device Security** > **Core Isolation Details** and make sure settings in this panel are turned **off**. A reboot is required to make this change.
  "Core isolation [includes] security features available on your device that use virtualization-based security"



Hardware components used to build this demonstration environment

- HP EliteBook 840 G5
- 32G of RAM
- Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
- System bios Q78 Ver. 01.09.01 - 10/16/2019
- SAMSUNG MZVLW512HMJP 512GB PCI Express 3.0 x4 (NVMe) SSD Drive.

Summary

The HPE Synergy demonstration environment installation is complete.

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

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

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

Happy demonstrations!

To help us improve this document, please provide feedback at lio@hpe.com.