neonHIVE Node Templates

for Ubuntu 16.04 LTS

Introduction

This document details the steps required to configure a Template VHDX for Hyper-V and an XVA suitable for hosting neonHIVEs for Hyper-V and XenServer.

VHDX Template Creation

Follow these instructions to create an Ubuntu-16.04 VHDX that can be used to quickly instantiate new Ubuntu VMs for development and test or to use to generate a PXE boot image for staging and production. Note that these steps will only be **rarely necessary** to regenerate the VHDX from scratch. Most of the time, you’ll be able to quickly **clone the pre-built VHDX** downloaded from Amazon S3 from (where **#** is the Ubuntu revision):

<https://s3.amazonaws.com/neonforge/neoncluster/neon-ubuntu-16.04.#.vhdx>

or the latest version from:  
  
 <https://s3.amazonaws.com/neonforge/neoncluster/neon-ubuntu-16.04.latest.vhdx>  
  
NOTE: These files are actually **ZIP archives**.

**Setup Instructions**

1. Download the **Ubuntu-16.04 Server ISO** from (where **#** is the desired revision):   
     
   <https://s3.amazonaws.com/neonforge/neoncluster/ubuntu-16.04.#-server-amd64.iso>
2. Open the **Hyper-V Manager** and step through the **New Virtual Machine Wizard**:  
   1. Name the VM **ubuntu-16.04-prep**.
   2. Configure **Generation 1**Note: I tried using generation 2 but I was unable to clone VMs by making copies of the template VHD when I did this. The cloned VMs wouldn’t boot.
   3. Then **1024MB RAM**.
   4. Set the networking connection to a switch with external access (e.g. **Intel® Ethernet Connection (2) I218- Virtual Switch** or **Default Switch)**
   5. Create a **10GB VHD** (the disk will be resized during hive provisioning). Locate this in a known directory where you’ll be able to find it. Name the disk:  
        
      **ubuntu-16.04-prep.vhdx**
   6. Installation Options: Configure to **boot** from the downloaded **Ubuntu ISO** and then press **Next/Finish**.
   7. Select the new VM in the Hyper-V manager, select **Settings**, click **Processor** in the left panel and set **4 Virtual Processors**.
   8. Recent versions of Windows 10 (as of August 2017) configure new VMs to **automatically checkpoint** their virtual hard drives (which is annoying). **Disable this** by selecting the **CheckPoints** folder in left settings panel (under **Management**) and **uncheck** **Enable checkpoints**.
   9. Press **OK** to save the settings.
3. **Start the VM** and then **double-click** to **connect** via the Hyper-V Manager.  
     
   You may see checkpoint error messages when you start the VM. I’m not entirely sure why this happens. I noticed that if I viewed the VM settings and then cancelled the dialog, the VM will start afterwards.
4. Press enter to select **English**.
5. Press enter to **Install Ubuntu Server**.
6. Press enter twice to select **English** and **United States**.
7. Press enter three times to select **English (US) keyboard**.
8. A bunch of stuff will be installed.
9. Make sure the host name is **ubuntu** and **Enter**.  
     
   \*\* **Do not change this** to avoid breaking subsequent scripts.
10. Create the **neon** **account:**Full Name: **sysadmin**  
    UID: **sysadmin**

PWD: **sysadmin0000**Note: These are the standard VM template credentials. A secure password will be set during hive setup.

1. **Do not encrypt the home directory**: Press **Enter** to skip.
2. Select **Yes** and configure the default **Time Zone** (a later script will change this to UTC).
3. Press **Up-Arrow** and then **Enter** to select **Guided – use entire disk** (**do not setup LVM!**).
4. Press **Enter** to accept the device changes.
5. Press **TAB** and **Enter** to select **Yes** to **Write the changes to disks** andconfigure **standard partitions** (not LVM).
6. Wait for the system to install.
7. Press Enter to **skip proxy** configuration.
8. More software will be installed.
9. Press **Enter** to select **No automatic updates** (I figure we’ll want to control when these happen).
10. Check **OpenSSH Server** in the **Software selection screen** by pressing **Space** and then press **Enter** to continue.
11. Even more software is installed.
12. Press **Enter** to Install the **GRUB** boot loader.
13. Press **Enter** to **Reboot**.
14. **Login** with the credentials you specified earlier to verify that the VM works.
15. Optional: Use the command below to discover the VM’s **IP address** for the **eth0** interface and connect via PuTTY or another terminal program that allows for easy copy and pasting of commands. Then connect to the server.

ip address

1. Run this command to start bash with root permissions:  
     
   sudo bash
2. Run the following command to modify **sudo** behavior so it doesn’t request passwords, making remote configuration possible:  
     
   echo "%sudo ALL=NOPASSWD: ALL" > /etc/sudoers.d/nopasswd
3. Run the following commands to install the ZIP package (required for neonHIVE setup):  
     
   apt-get update  
   apt-get install -yq zip
4. Install the Hardware Enablement (HWE) kernel and related daemons to support better integration with virtualization:  
     
   apt-get install -yq linux-virtual-lts-xenial linux-tools-virtual-lts-xenial linux-cloud-tools-virtual-lts-xenial
5. Ubuntu setup configures the disk with three partitions: the boot file system, a small extended partition and a swap partition. We need to disable swap and delete the two partitions after the filesystem so when we create a VM with a larger drive, we’ll be able to grow the filesystem to fill it. Here are the steps:  
   1. Edit /etc/fstab and **delete** the two **swap lines** near the end.
   2. Disable swap running: swapoff -a
   3. Run parted and remove the last two partitions:  
        
      rm 2  
      quit  
        
      Note: Removing the extended partition (#2) also removes the swap partition (#5).
6. Run the following command to clear cached packages and the **cached DHCP** leases and shutdown:  
     
   apt-get cleanrm -rf /var/lib/dhcp/\*  
   shutdown -h now
7. **Hyper-V Template Upload:** ZIP the disk and then use the AWS Console to **Upload** the image to the location below (where **#** is the revision) and grant **public read access**. Note that you need to ZIP the image using the **neon-cli** because the Windows file system ZIP feature uses Deflate64 compression which is not compatible with **neon-cli** (and other tools). Be patient; zipping is very slow:

neon zip create PATH-TO-VHDX PATH-TO-ZIP

Then upload the ZIP file to (**removing the .zip** extension):  
  
<https://s3.amazonaws.com/neonforge/neoncluster/neon-ubuntu-16.04.#.vhdx>

…and if this is the latest Ubuntu image, restart and log back into the VM and then run:  
  
sudo bash  
apt-get update  
apt-get dist-upgrade -yq  
apt-get clean  
rm -rf /var/lib/dhcp/\*  
shutdown -h now

and then re-ZIP it and upload to (**removing the .zip** extension):  
  
<https://s3.amazonaws.com/neonforge/neoncluster/neon-ubuntu-16.04.latest.vhdx>   
  
**Be sure to make these files public on AWS!**

# XenServer Template

**XenServer Template Upload:** XenServer templates are GZIP compressed rather than using ZIP so the XenServer tooling can download them. The steps for creating a XenServer image is like those for Hyper-V above. Essentially, you’ll use XenCenter to create the VM by mounting the Ubuntu setup ISO, setting up the VM, installing the XenServer tools, and then exporting it as an **\*.xva** file.  
  
I generally go through the full template build from ISO only for Ubuntu point releases. For intermediate package upgrades, I do the following:  
  
\* In XenCenter, rename the most recent Ubuntu point release template from **neon-ubuntu-16.04-template** to something like **x-neon-template**.  
  
\* Create a VM named **neon-ubuntu-16.04-template** from the **x-neon-template** template and run these commands to upgrade it:   
  
apt-get update  
apt-get dist-upgrade -yq  
rm -rf /var/lib/dhcp/\*  
shutdown -h now  
  
\* Convert the VM into a Xen Template and download/upload it to S3 as described further below.  
  
\* Delete the **neon-template** VM and then **rename x-neon-template** back to **neon-ubuntu-16.04-template**.  
  
Follow the steps below compress and upload it to AWS S3:

* 1. Create and initialize an Ubuntu virtual machine on XenServer much like we did above for Hyper-V. Name the VM **neon-ubuntu-16.04-template (this is important)** You’ll be using the console in XenCenter.
  2. After you **finished installing Ubuntu**: **right-click** on the VM in XenCenter and click **Install XenServer Tools**. You may need to wait a minute or two for the menu item to show up.
  3. **Mount** the tools DVD drive:  
       
     sudo bash  
     mount -o ro,exec /dev/sr0 /mnt  
       
     You may need to modify the /dev/sr0 path. Execute blkid and use the device with the XenServer Tools label. You may need to wait a minute or two for the DVD to mount.
  4. **Install** the tools via and then reboot:  
       
     /mnt/Linux/install.sh  
     reboot
  5. **Login** the VM.
  6. **Eject** the **DVD drive** in XenCenter (at the top-right of the VMs storage tab) and shutdown the VM:  
       
     rm -rf /var/lib/dhcp/\*  
     shutdown -h now
  7. Right-click the VM and **Convert to Template**.
  8. Right-click on the template and select **Export to file…** Save the file as **neon-ubuntu-16-04.#.xva** where **#** is the revision, set the **format to XVA** and then export to a local file.
  9. GZIP compress the file and create a latest version if this is the latest. Note that we’re going to rename the file to remove the **.gz** file type after uploading to AWS:  
       
     gzip --best neon-ubuntu-16.04.4.xva
  10. If this is the latest Ubuntu image, restart and log back into the VM and then run:  
        
      apt-get update  
      apt-get dist-upgrade -yq  
      rm -rf /var/lib/dhcp/\*  
      shutdown -h now  
        
      After the VM has down, export it to **neon-ubuntu-16-04.latest.xva** and then GZIP it like:  
        
      gzip --best neon-ubuntu-16.04.latest.xva

1. Manually upload the files to S3 setting metadata to:  
     
   **S3** at [https://s3.amazonaws.com/neonforge/neoncluster/\*](https://s3.amazonaws.com/neonforge/neoncluster/*)   
     
   Be sure to: **Remove the .gz extension.**Be sure to: **Add AWS metadata: Content-Encoding = gzip**Be sure to: **Make these files public on AWS!**