Ubuntu 18.04 Template for Hyper-V

Introduction

This document details the steps required to configure a template VHDX for Hyper-V.

VHDX Template Creation

Follow these instructions to create an Ubuntu-18.04 VHDX that can be used to quickly instantiate new deploy Kubernetes to Hyper-V. 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/kube/ubuntu-18.04.#.vhdx>

or the latest version from:  
  
 <https://s3.amazonaws.com/neonforge/kube/ubuntu-18.04.latest.vhdx>  
  
NOTE: These files are **GZIP** encoded…

**Setup Instructions**

1. Download the **Ubuntu-18.04 Server ISO** from (where **#** is the desired revision) from:   
     
   http://releases.ubuntu.com/18.04/

Edit the downloaded file name to remove the last version field (probably “.0”) and the “live-server-amd64” and then archive the ISO file to S3 (making it public):  
  
<https://s3.amazonaws.com/neonforge/kube/ubuntu-18.04.#.iso>

1. Create the VHDX we’ll use for the VM via PowerShell:  
     
   powershell New-VHD -Path D:\VM\ubuntu-18.04.prep.vhdx -SizeBytes 10GB -Dynamic -BlockSizeBytes 1MB
2. Open the **Hyper-V Manager** and step through the **New Virtual Machine Wizard**:  
   1. Name the VM: **ubuntu-18.04.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.
   5. Use the VHDX created above for the new VM:  
        
      **ubuntu-18.04.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. Wait for the installation UX to start.
5. Press enter to select **English**.
6. Press enter to **Install**.
7. Press enter to skip proxy configuration.
8. Press enter to choose the default mirror.
9. Press enter three times to use the entire disk.
10. Press the down arrow and enter to confirm the file system settings.
11. Enter **sysadmin** as your name and username and enter **sysadmin0000** as the password. Set the server name to **ubuntu** and select **done**.
12. Don’t install any server Snaps.
13. Start the installation.
14. Select **Reboot Now**.
15. **Login** with the credentials you specified earlier to verify that the VM works.
16. 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 (the password is **sysadmin0000**):  
     
   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 some required packages:  
     
   apt-get update  
   apt-get install -yq zip secure-delete
4. Disable swap by editing /etc/fstab and removing the /swap.img line.
5. Configure the Linux guest integration services:  
   1. nano /etc/initramfs-tools/modules
   2. Append this the file and save:  
        
      hv\_vmbus  
      hv\_storvsc  
      hv\_blkvsc  
      hv\_netvsc
   3. Install and configure the integration services and reboot:  
        
      apt-get install -yq linux-virtual linux-cloud-tools-virtual linux-tools-virtual  
      update-initramfs -u  
      reboot
6. Run the following command to clear cached packages and the **cached DHCP** leases and then, zero deleted files (for better compression) and shutdown:  
     
   apt-get cleanrm -rf /var/lib/dhcp/\*  
   sfill -fllz /  
   shutdown -h now
7. **Hyper-V Template Upload:** Copy the VHDX somewhere else and GZIP it:  
     
   gzip --best PATH-TO-VHDX  
     
   Then use the AWS Console to **Upload** the image to the location below (where **#** is the revision) and grant **public read access**:

Then upload the ZIP file to S3 (**removing the .gz** extension and setting **Content-Encoding=gzip**):  
  
<https://s3.amazonaws.com/neonforge/kube/kube-ubuntu-18.04.#.vhdx>

…and if this is the latest Ubuntu image, restart and log back into the VM and then run these commands to fully **upgrade the image** (if you see a **lock error**, restart and wait a bit before retrying):  
  
sudo bash

apt-get update

apt-get dist-upgrade -yq  
  
…and then these commands to **clean the disk** and **shutdown**:

apt-get clean

rm -rf /var/lib/dhcp/\*  
sfill -fllz /

shutdown -h now

and compress it and upload to (**removing the .gz** extension and setting **Content-Encoding=gzip**):  
  
<https://s3.amazonaws.com/neonforge/kube/kube-ubuntu-18.04.latest.vhdx>   
  
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!**