Ubuntu 20.04 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-20.04 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-20.04.#.vhdx>

or the latest version from:  
  
 <https://s3.amazonaws.com/neonforge/kube/ubuntu-20.04.latest.vhdx>  
  
NOTE: These files are compressed via **GZIP**.

**Setup Instructions**

1. Download the **ubuntu-20.04-live-server-amd64.iso** from (where **#** is the desired revision) from:   
     
   <http://releases.ubuntu.com/20.04/>

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-20.04.#.iso>

1. Create the VHDX we’ll use for the VM via PowerShell (**change the “# “ to match the Ubuntu version**):  
     
   powershell New-VHD -Path D:\VM\hyperv-ubuntu-20.04.#.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-20.04.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-20.04.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. **DO NOT UPDATE** to the **latest installer** if asked.
7. Press enter to **Install**.
8. Press enter to **skip proxy** configuration.
9. Press enter to choose the **default mirror**.
10. **Use the entire disk**.
11. **Confirm the file system settings**.
12. **Continue with setup** (there’s no going back 😊)
13. Enter **sysadmin** as your **name** and **username** and enter **sysadmin0000** as the password. Set the server name to **ubuntu** and select **done**.
14. Install **OpenSSH**.
15. Install **NO Server Snaps**.
16. Let the installation get to the **security update** step and then **cancel the update and reboot** (the idea is that we want our templates to match the Ubuntu releases and that we’ll handle updates separately).
17. Select **Reboot Now** and then press **ENTER** to reboot (you may see a failed CDROM eject message: ignore that and just press **ENTER**).
18. **Login** with the credentials you specified earlier to verify that the VM works.
19. Use the command below to discover the VM’s **IP address** for the **eth0** interface. You’ll need this later to finish preparing the node.

ip address

1. **IMPORTANT:** Be sure to **logout** of any PuTTY or Hyper-V terminals so the next step will work.
2. Run this command **on your workstation** to complete the preparation, passing the IP address of the VM:  
     
   neon prepare node-template --hyperv IPADDRESS
3. **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/hyperv-ubuntu-20.04.#.vhdx>

…and if this is the latest Ubuntu release, restart and log back into the VM and then run these commands to fully **upgrade the template** (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/hyperv-ubuntu-20.04.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!**