

Boîte de câbles

SOUS-SYSTÈME WINDOWS POUR LINUX

Invités KVM accélérés sur WSL 2

Internet l'a demandé.



Hayden Barnes

14 juin 2020 • 10 minutes de lecture



Ce guide est obsolète au profit de cette mise à jour 2022.

Exigences

Windows 10 Insider Fast Ring, testé sur la build 19645.1

Ubuntu 20.04 sur WSL 2

Un processeur Intel Core i5+*

Un serveur X pour Windows, par exemple X410 ou VcXsrv (Pour l'instant, le support GUI natif et l'accélération GPU arrivent !)

Au moins 16 Go de RAM

À au moins 40 Go d'espace libre sur le disque dur

*Utilisateurs AMD : WSL 2 s'exécute sur une plate-forme Hyper-V légère sur n'importe quelle édition de Windows 10. Hyper-V vient de bénéficier de la prise en charge de la virtualisation imbriquée AMD. Ils ont annoncé que le support invité Linux KVM arriverait.

Recommandé

Familiarité avec la création d'un noyau Linux**

Familiarité avec KVM, QEMU ou la technologie de virtualisation

Terminal Windows (Obtenez-le. C'est rapide !)

**Si vous n'êtes pas prêt à créer des modules du noyau Linux mais que vous souhaitez commencer, consultez mon guide sur l'exécution de Windows 2000 avec QEMU sur WSL.

Mettre à jour, mettre à niveau et installer les dépendances

Vous avez régulièrement mis à jour votre distribution WSL, n'est-ce pas ?

```
udo apt -y upgrade  
build-essential libncurses-dev bison flex libssl-dev libelf-dev cpu-checker qemu-kvm a
```

Récupérer les sources du noyau

Vous avez besoin d'un noyau. Vous pouvez utiliser n'importe quel noyau, y compris celui de votre distribution, mais pour plus de simplicité, j'utilise le terminal Microsoft WSL par défaut comme base.

Il existe peut-être des noyaux WSL 2 plus récents disponibles, mais ils n'ont été testés que sur la version 4.19.104 (et vous aurez besoin de la version 4.19.121 ou supérieure pour le calcul GPU) :

```
cd ~  
aria2c -x 10 https://github.com/microsoft/WSL2-Linux-Kernel/archive/4.19.104-microso  
tar -xf WSL2-Linux-Kernel-4.19.104-microsoft-standard.tar.gz  
cd WSL2-Linux-Kernel-4.19.104-microsoft-standard/
```

Si vous lisez ceci à l'avenir, vous pouvez vérifier votre numéro de version WSL 2 actuel avec :

```
uname -r
```

Vous pouvez trouver toutes les versions du noyau WSL 2 sur [GitHub](#).

Vous pouvez également utiliser git pour récupérer des branches spécifiques et des versions balisées du noyau WSL 2.

Optimiser la configuration du noyau pour KVM sur WSL

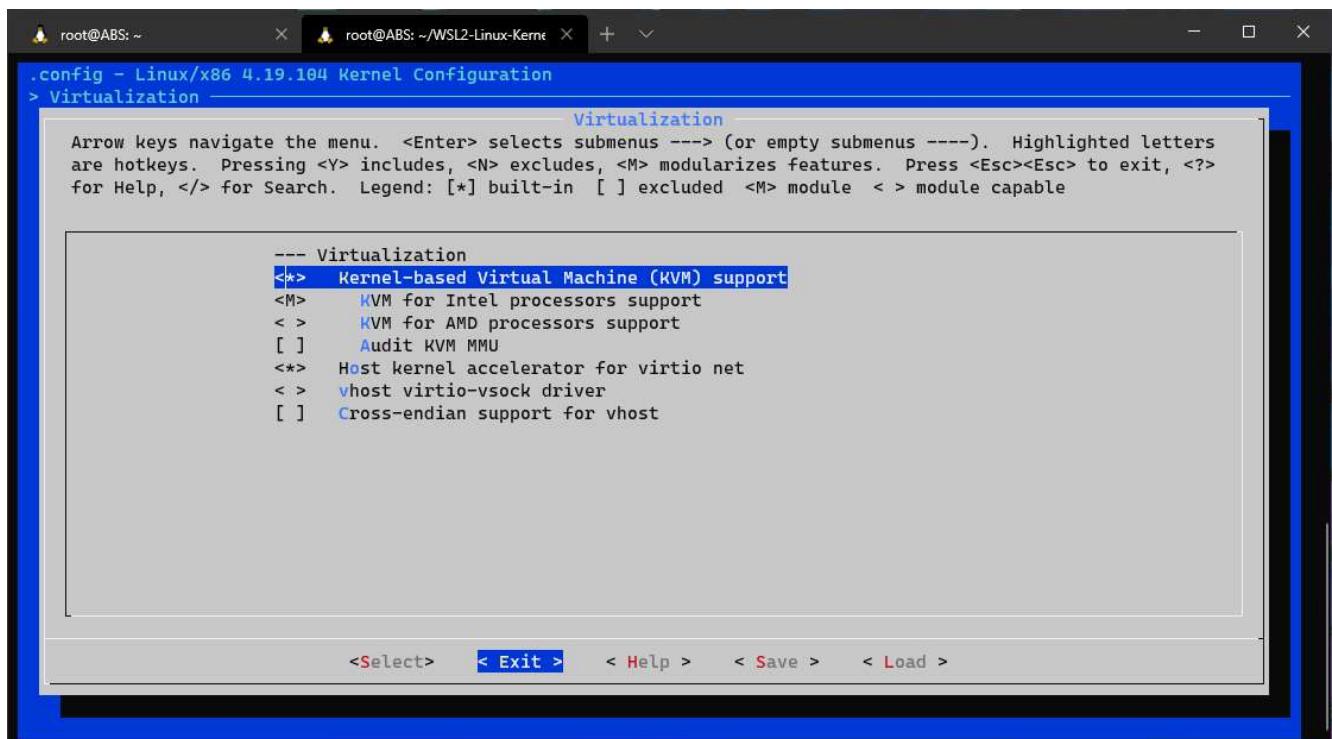
Commencez par emprunter la configuration officielle du noyau WSL 2 :

```
cp Microsoft/config-wsl .config
```

Alors personnalisons-le :

```
make menuconfig
```

Sous Virtualisation, j'ai configuré KVM pour la prise en charge des processeurs Intel pour qu'il soit construit en tant que module (lisez ci-dessous et décidez si vous devez le faire) et j'active la prise en charge virtio net intégrée :



Remarque importante sur la construction d'un module en WSL :

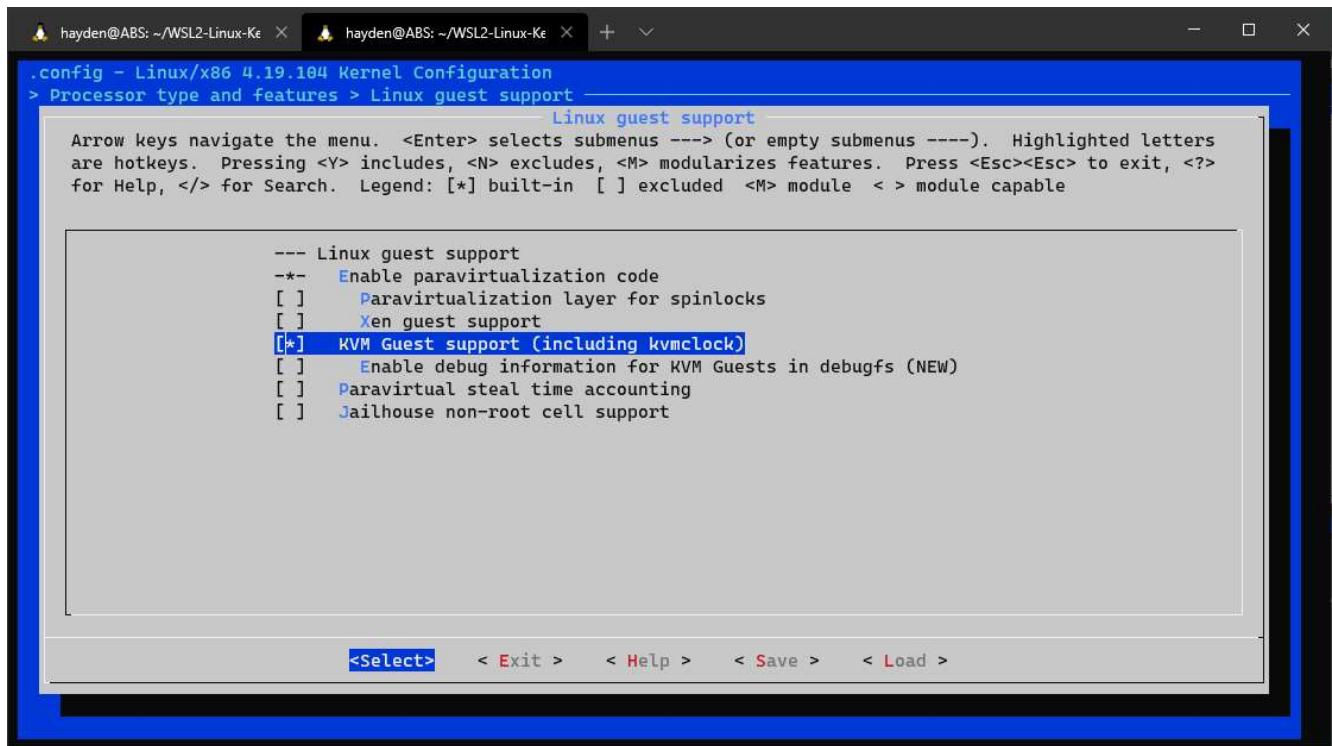
La création de KVM pour Intel en tant que module <M> facilite le déchargement, la modification et le rechargement du module selon les besoins, par exemple :

```
sudo modprobe -r kvm_intel # removes the module  
modprobe kvm_intel enable_apicv=0 # reloads the module with apicv disable
```

Cela peut être utile pour que certains systèmes d'exploitation s'exécutent ou affinent leurs performances. **Mais** cela signifie que vous devrez charger le module à chaque fois manuellement ou l'ajouter à votre .bashrc.

Si vous ne souhaitez pas faire cela, laissez simplement la prise en charge de KVM pour les processeurs Intel comme <*> intégrée, et vous pouvez ignorer l'étape make modules_install ci-dessous, mais assurez-vous de toujours ajouter la prise en charge des invités KVM et de configurer kvm_intel dans /etc/modprobe.d/kvm-nested.conf comme décrit ci-dessous.

Exit the Virtualization menu and then proceed to Processor type and features -> Linux guest support enable built-in KVM guest support:



Exit and save as .config.

Flags

If you are 10x and too cool to use a GUI, you can edit your .config directly in vim:

```
KVM_GUEST=y
CONFIG_KVM=y
CONFIG_KVM_INTEL=m
CONFIG_VHOST_NET=y
CONFIG_VHOST=y
```

Build the new kernel

```
make -j 8
```

Sit back and watch a movie while you wait:



Install your kernel modules

```
sudo make modules_install
```

Install your kernel in WSL 2 and enable nested KVM

```
cp arch/x86/boot/bzImage /mnt/c/Users/<username>/bzImage  
nano /mnt/c/Users/<username>/.wslconfig
```

Paste but don't forget to change <username> to your Windows username:

```
[wsl2]  
nestedVirtualization=true  
kernel=C:\\Users\\<username>\\bzImage
```

WSL 2 Settings Pro Tip

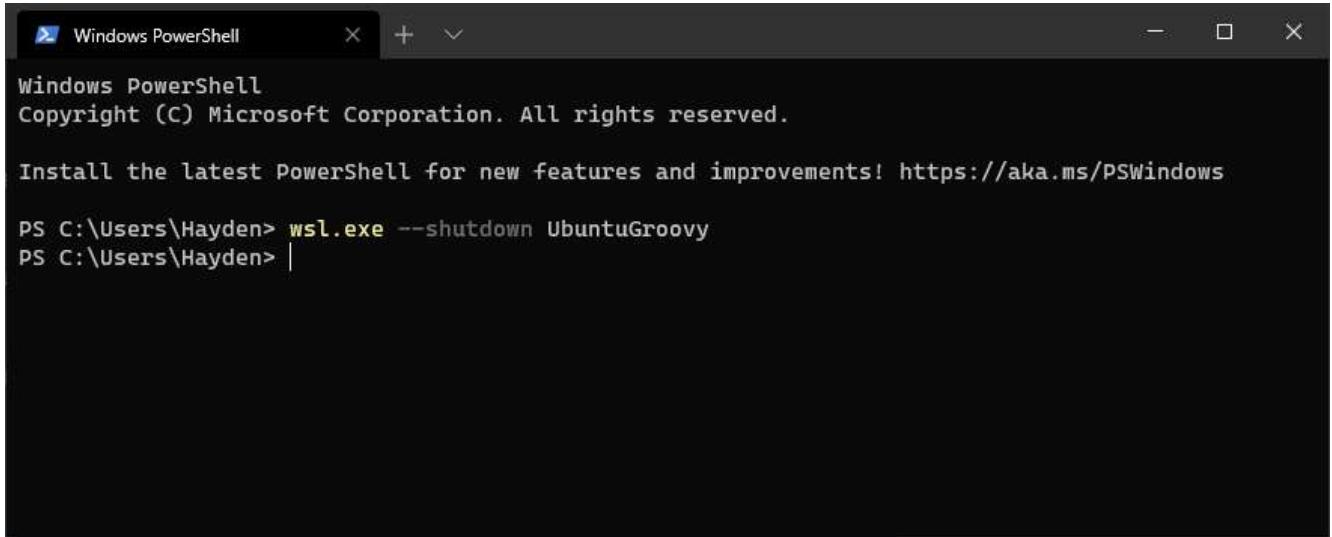
- /etc/wsl.conf manages your distro-specific WSL settings
- %HOMEPATH%\wslconfig manages for global WSL settings

Reboot WSL 2

Close all open WSL instances. If you are using Windows Terminal just open a PowerShell terminal tab and close the other tabs. See, I told you to get Windows Terminal.

In a new PowerShell terminal:

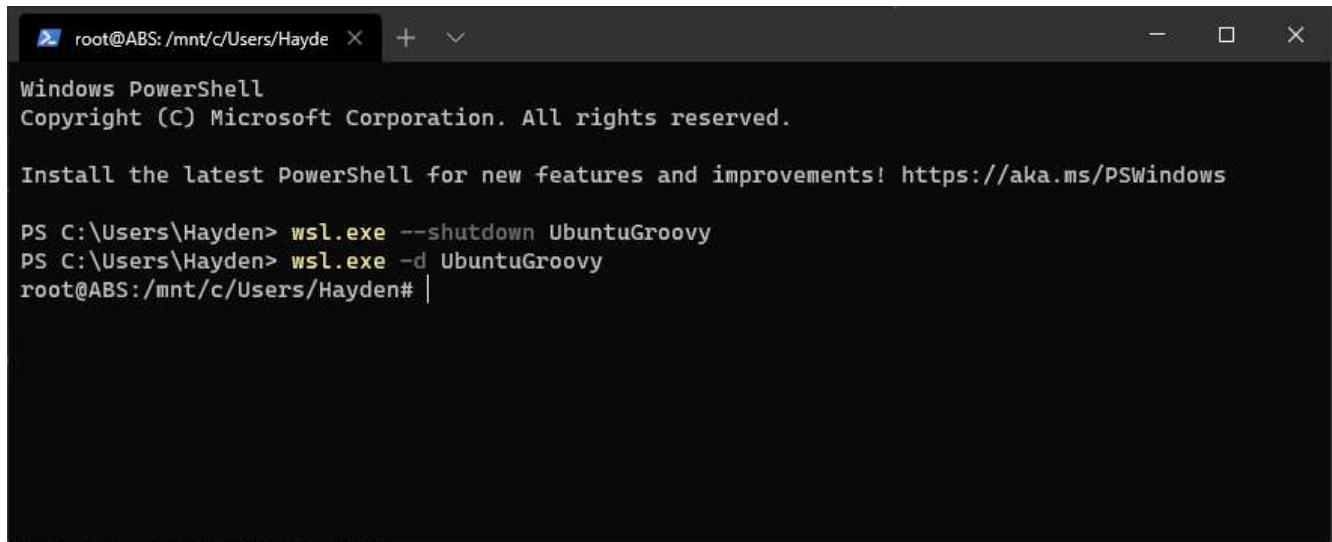
```
wsl.exe --shutdown Ubuntu
```



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the following text:
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>
PS C:\Users\Hayden> wsl.exe --shutdown UbuntuGroovy
PS C:\Users\Hayden> |

Re-open Ubuntu on WSL:

```
wsl.exe -d Ubuntu
```



```
root@ABS:/mnt/c/Users/Hayde X + 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

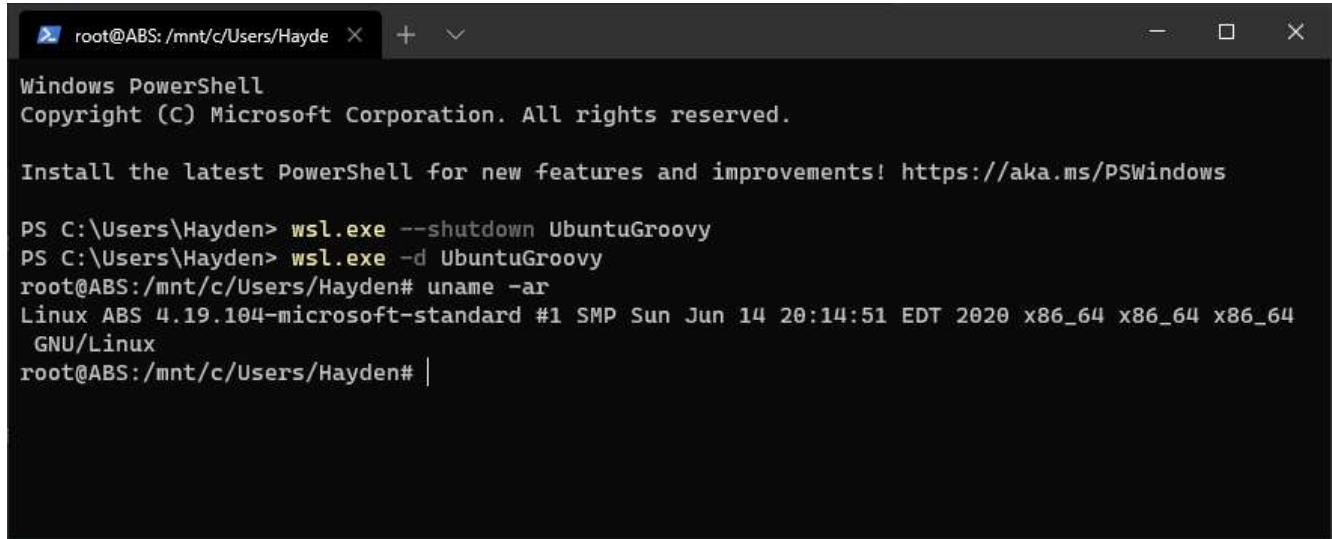
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hayden> wsl.exe --shutdown UbuntuGroovy
PS C:\Users\Hayden> wsl.exe -d UbuntuGroovy
root@ABS:/mnt/c/Users/Hayden# |
```

Check you are running your new kernel

```
uname -ar
```

should show today's date and the time should be a few minutes ago.



```
root@ABS:/mnt/c/Users/Hayde X + 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hayden> wsl.exe --shutdown UbuntuGroovy
PS C:\Users\Hayden> wsl.exe -d UbuntuGroovy
root@ABS:/mnt/c/Users/Hayden# uname -ar
Linux ABS 4.19.104-microsoft-standard #1 SMP Sun Jun 14 20:14:51 EDT 2020 x86_64 x86_64 x86_64
GNU/Linux
root@ABS:/mnt/c/Users/Hayden# |
```

Configure kvm-intel

```
nano /etc/modprobe.d/kvm-nested.conf
```

Paste:

```
options kvm-intel nested=1
options kvm-intel enable_shadow_vmcs=1
options kvm-intel enable_apicv=1
options kvm-intel ept=1
```

GNU nano 4.8 /etc/modprobe.d/kvm-nested.conf Modified

```
options kvm-intel nested=1
options kvm-intel enable_shadow_vmcs=1
options kvm-intel enable_apicv=1
options kvm-intel ept=1
```

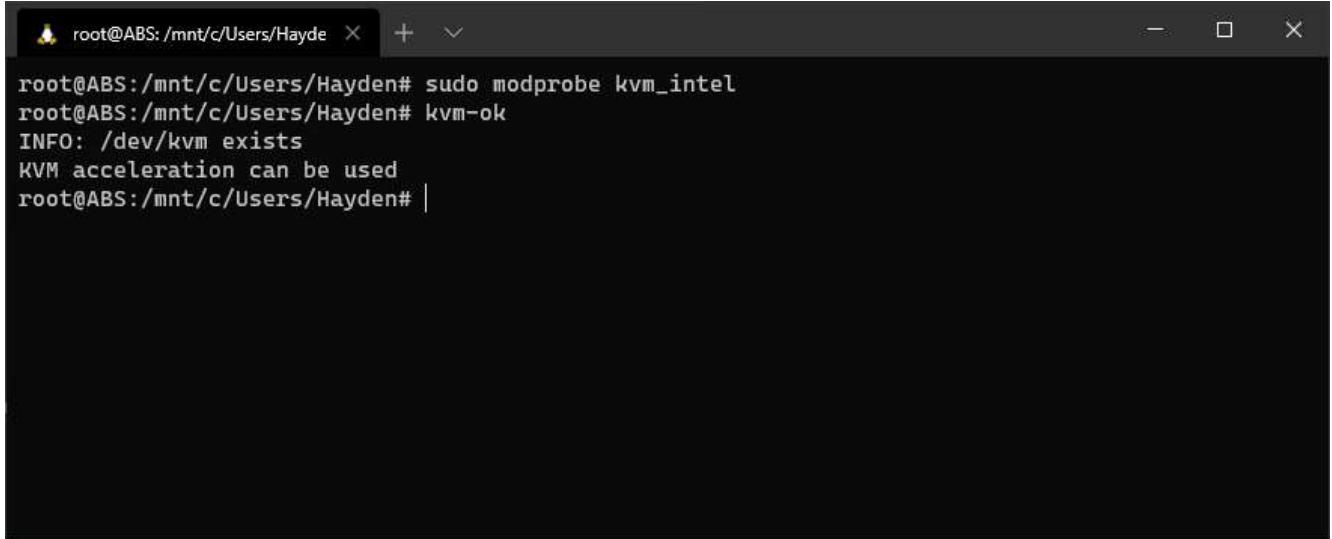
Save modified buffer? |
Y Yes
N No ^C Cancel

Load kernel module

```
sudo modprobe kvm_intel
```

Test KVM with:

```
kvm-ok
```

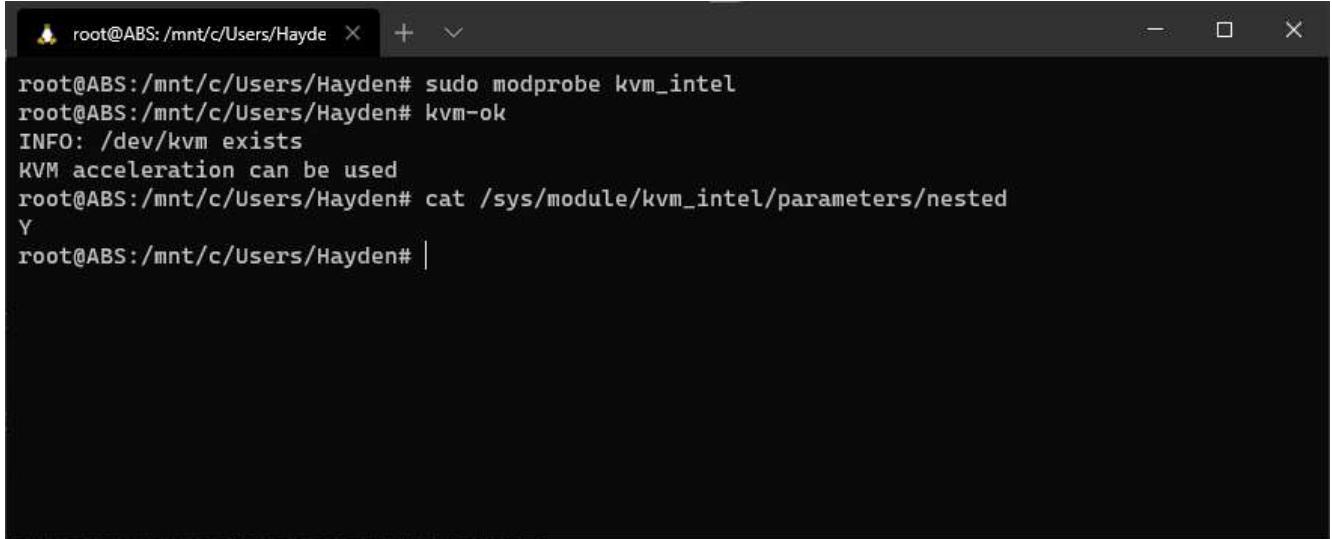


```
root@ABS:/mnt/c/Users/Hayde# sudo modprobe kvm_intel
root@ABS:/mnt/c/Users/Hayde# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
root@ABS:/mnt/c/Users/Hayde# |
```

Check for nested KVM with:

```
cat /sys/module/kvm_intel/parameters/nested
```

should report "Y"



```
root@ABS:/mnt/c/Users/Hayde# sudo modprobe kvm_intel
root@ABS:/mnt/c/Users/Hayde# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
root@ABS:/mnt/c/Users/Hayde# cat /sys/module/kvm_intel/parameters/nested
Y
root@ABS:/mnt/c/Users/Hayde# |
```

Configure X for WSL 2

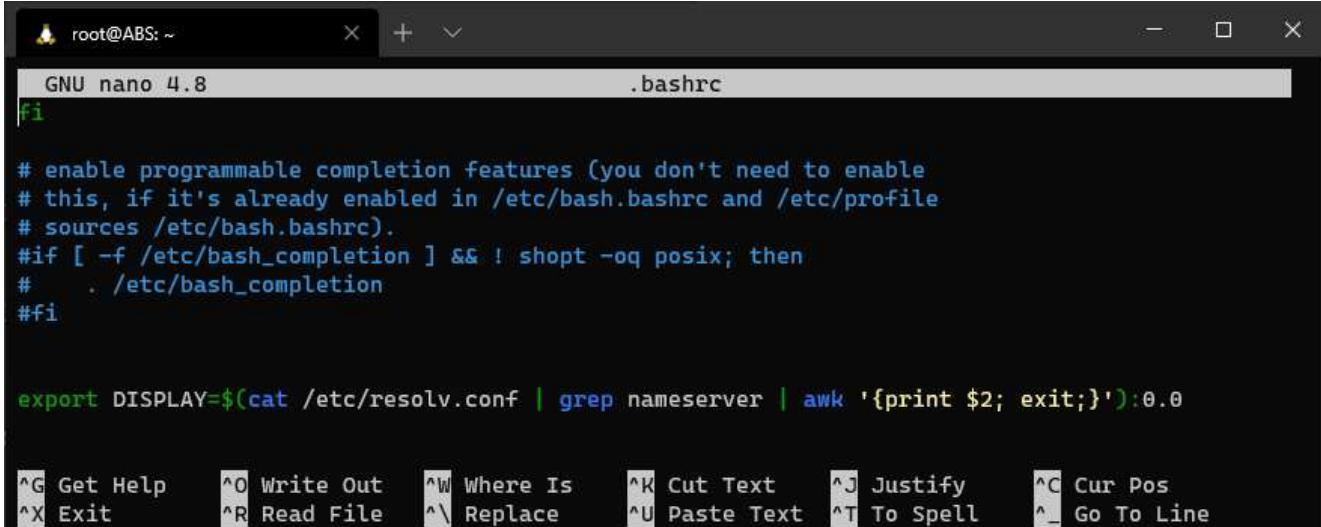
Install X410 or VcXsrv and configure firewall settings to allow on Public network. I prefer X410 because it does it for you, but it costs

about \$10 USD in the Store. VcXsrc is free and open source but requires a bit more manual configuration Native GUI support is coming, more details later in 2020.

Once you are running X on Windows, redirect your WSL X output to Windows by setting the DISPLAY environmental variable:

```
export DISPLAY=$(cat /etc/resolv.conf | grep nameserver | awk '{print $2; exit;}'):#0.0
```

You can add this command to `~/.bashrc` to run every time you open Ubuntu WSL:



```
GNU nano 4.8 .bashrc
#i

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#   . /etc/bash_completion
#fi

export DISPLAY=$(cat /etc/resolv.conf | grep nameserver | awk '{print $2; exit;}'):#0.0

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify     ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell    ^_ Go To Line
```

If you add this command to your `.bashrc` file, then to load it for this session just:

```
source ~/.bashrc
```

Download an Ubuntu desktop install ISO

Go back to your home folder, or make a folder inside WSL to keep this project (NOT on /mnt/c/ it will slow you down), and then download the nightly Ubuntu desktop install ISO with aria2:

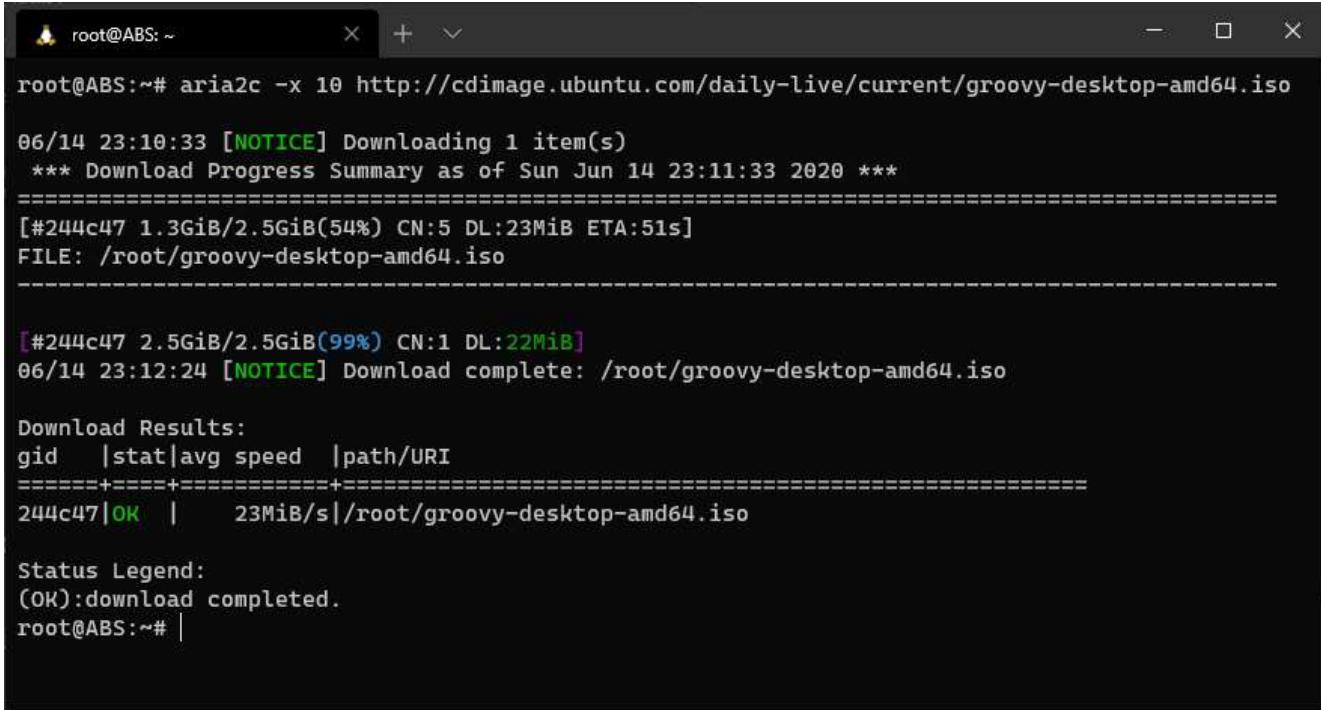
```
cd ~  
aria2c -x 10 http://cdimage.ubuntu.com/daily-live/current/groovy-desktop-amd64.iso
```

You can get ISOs for Ubuntu 20.04 LTS (focal) and many of the official flavors [here](#).

aria2

aria2 is a next generation download utility.

Use aria2c in place of wget for all your downloads, add -x 10 to break the download up into 10 download threads, it speeds things up quite a bit.

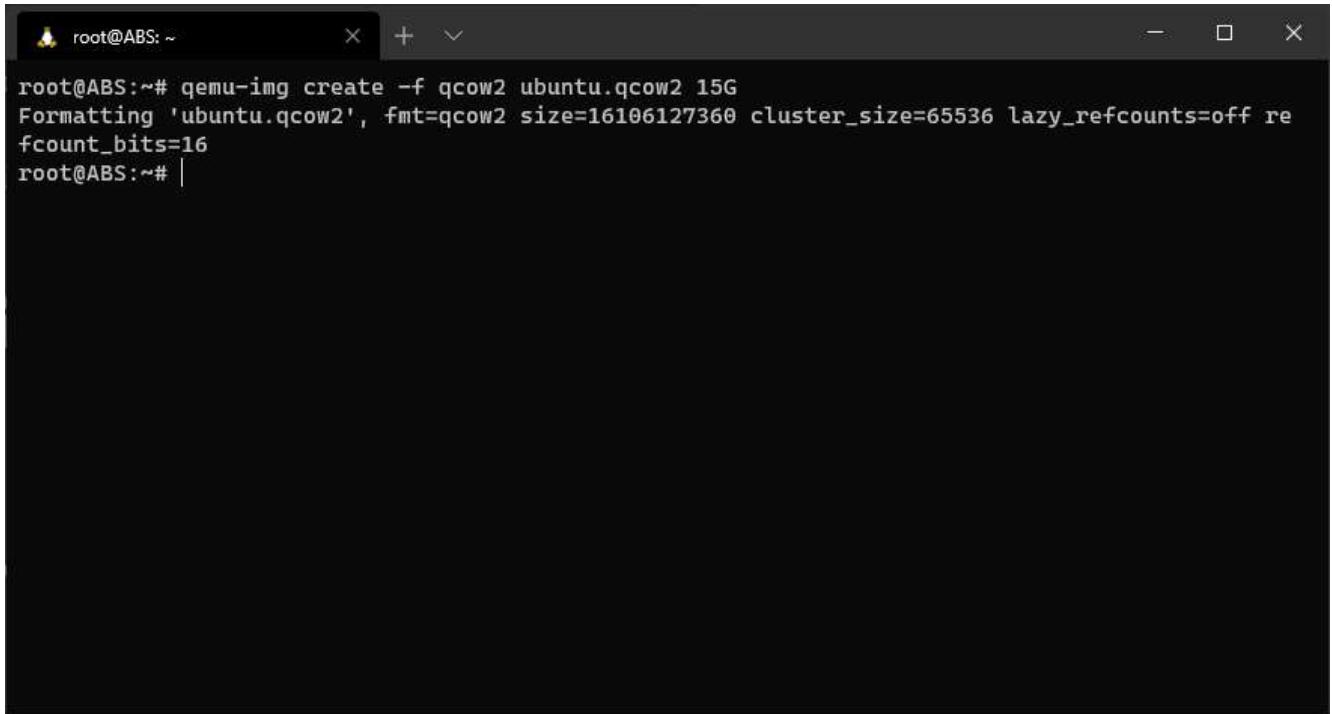


```
root@ABS:~# aria2c -x 10 http://cdimage.ubuntu.com/daily-live/current/groovy-desktop-amd64.iso  
06/14 23:10:33 [NOTICE] Downloading 1 item(s)  
*** Download Progress Summary as of Sun Jun 14 23:11:33 2020 ***  
===== [#244c47 1.3GiB/2.5GiB(54%) CN:5 DL:23MiB ETA:51s]  
FILE: /root/groovy-desktop-amd64.iso  
===== [#244c47 2.5GiB/2.5GiB(99%) CN:1 DL:22MiB]  
06/14 23:12:24 [NOTICE] Download complete: /root/groovy-desktop-amd64.iso  
  
Download Results:  
gid |stat|avg speed |path/URI  
===== [#244c47|OK| 23MiB/s|/root/groovy-desktop-amd64.iso]  
  
Status Legend:  
(OK):download completed.  
root@ABS:~#
```

Create a qcow2 virtual hard drive for QEMU

This will be our virtual hard drive for our Ubuntu . qcow2 is the default virtual hard drive container for QEMU, similar to .vhd, .vdi, .vmddk files.

```
qemu-img create -f qcow2 ubuntu.qcow2 15G
```

A screenshot of a terminal window titled "root@ABS: ~". The window shows a command being run: "qemu-img create -f qcow2 ubuntu.qcow2 15G". The output of the command is displayed below the command line, indicating that a new qcow2 file named "ubuntu.qcow2" is being created with a size of 15G. The terminal window has a dark background and light-colored text. The title bar includes the user name "root@ABS" and the current directory "~".

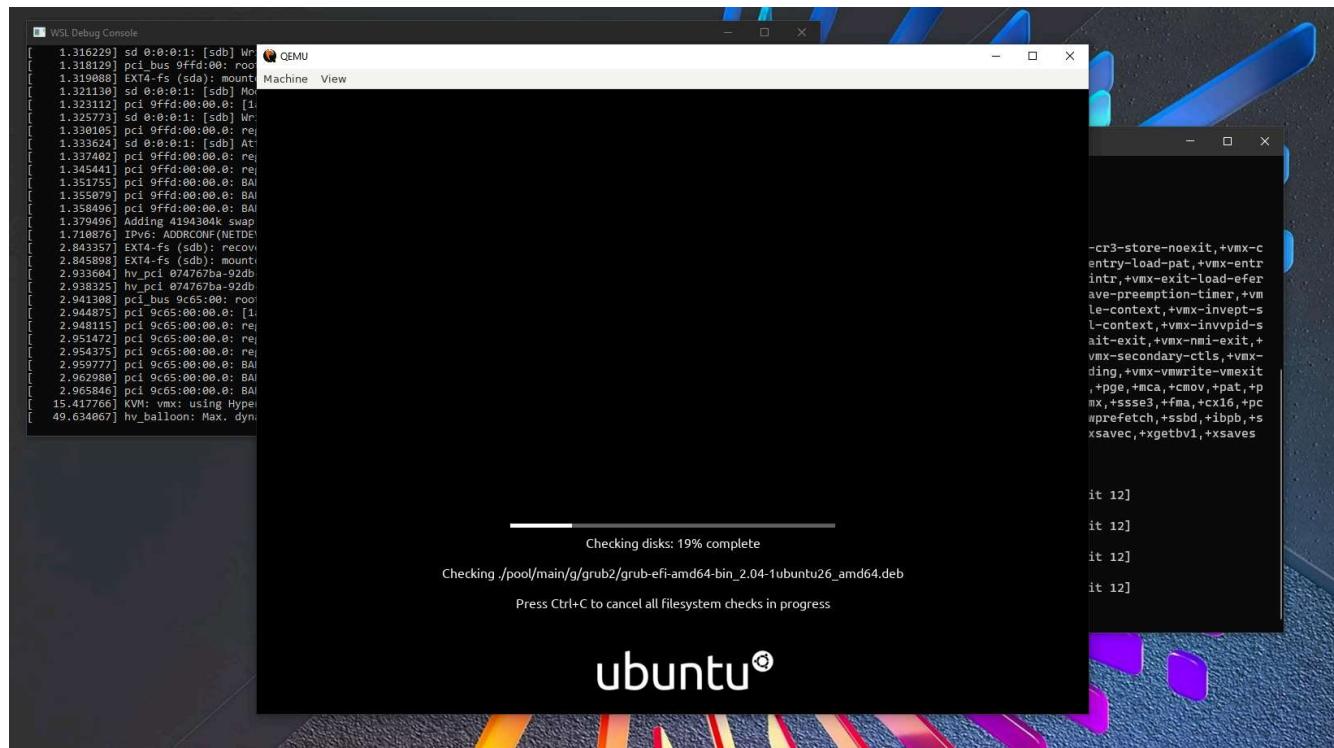
```
root@ABS:~# qemu-img create -f qcow2 ubuntu.qcow2 15G
Formatting 'ubuntu.qcow2', fmt=qcow2 size=16106127360 cluster_size=65536 lazy_refcounts=off refcount_bits=16
root@ABS:~# |
```

The Ubuntu installer will detect the virtual drive, partition, install, and install a bootloader. If you are installing another distro or operating system you may need to manually format and partition as part of the install process.

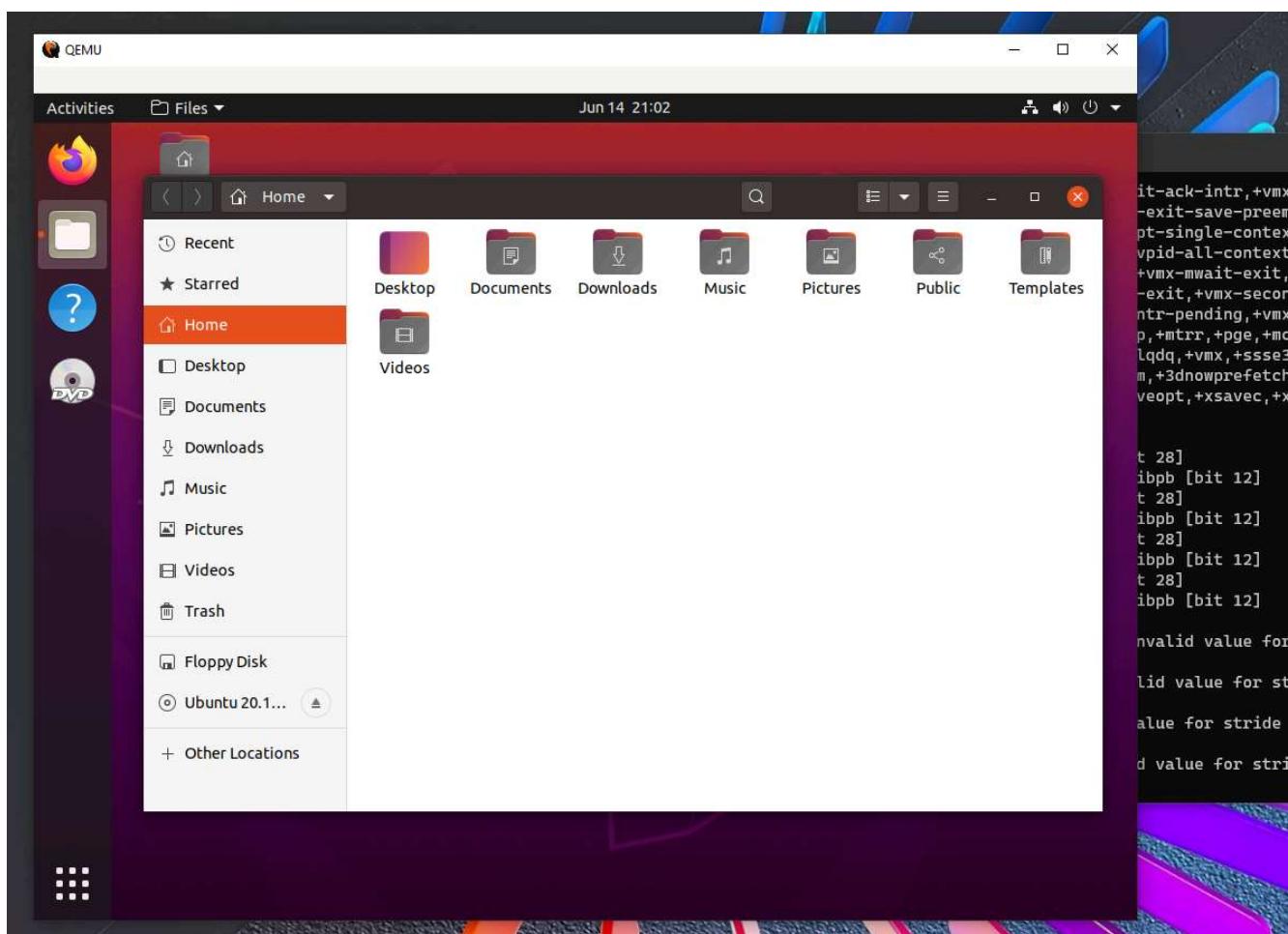
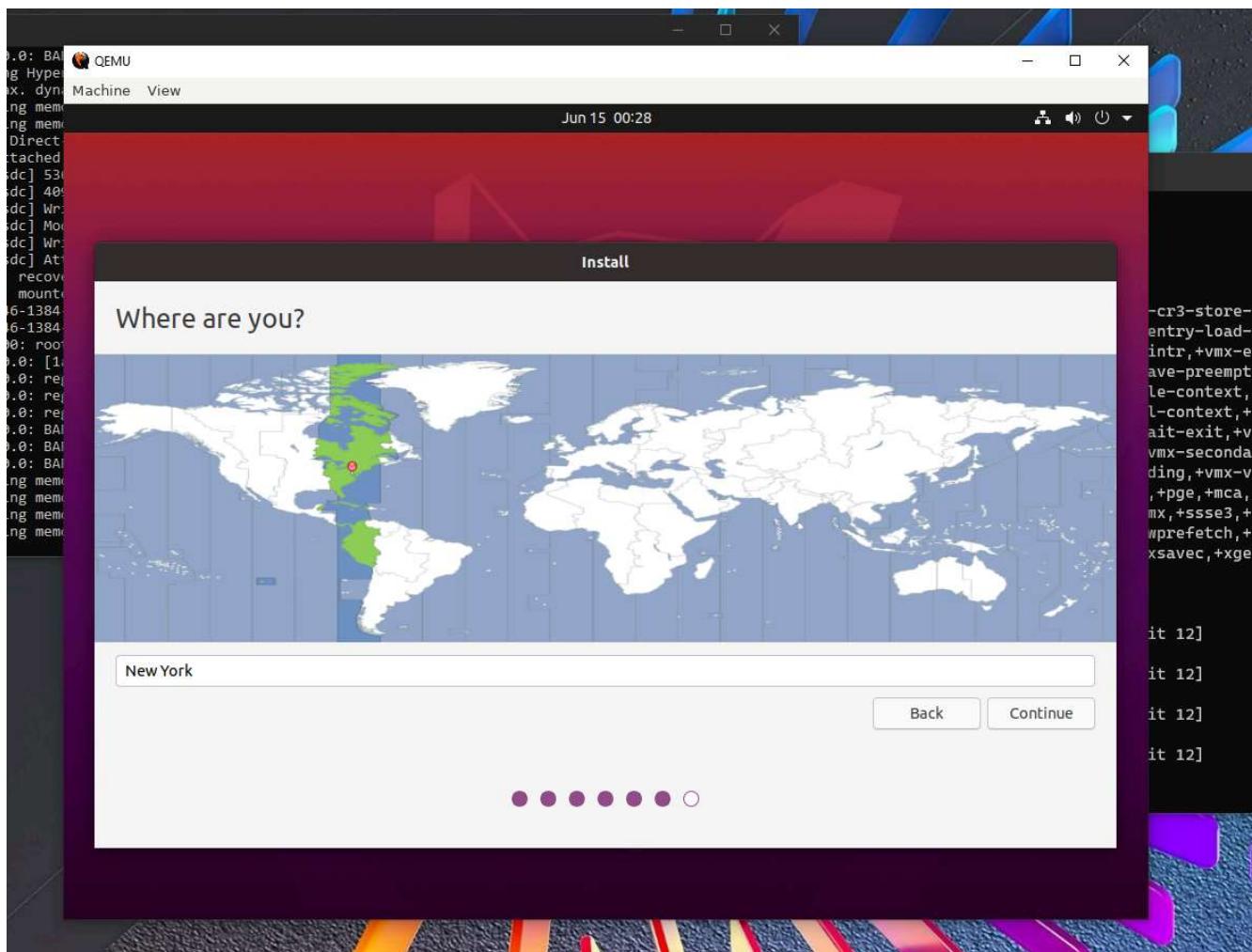
Boot from the ISO and install to the virtual hard drive

[Pastebin](#)

```
qemu-system-x86_64 \
    -drive file=ubuntu.qcow2,format=qcow2 \
    -drive file=groovy-desktop-amd64.iso,media=cdrom,readonly \
    -net nic -net user \
    -m 5172 \
    -vga qxl \
    --enable-kvm \
    -smp 4 \
    -cpu kvm64,+vmx,+vme,+msr,+x2apic,+hypervisor,+aes,+vmx-activity-hlt,+vmx-cr
```



Then follow normal Ubuntu installation steps:



What is all this?

Cela lance QEMU, un émulateur/virtualiseur de système open source. QEMU est un outil puissant que j'utilise souvent. L'une de mes utilisations préférées consiste à émuler du matériel vintage pour exécuter des systèmes d'exploitation vintage. J'ai écrit un guide pour exécuter Windows 2000 sur WSL à l'aide de QEMU.

La plupart des indicateurs, comme le montage du disque dur virtuel et l'ISO en tant que cdrom, sont explicites.

Pour en savoir plus, consultez la page de manuel de QEMU :

[qemu-system\(1\) — qemu-system-common — Tests Debian](#)
[— Pages de manuel Debian](#)

Il y a plus de liens vers des documents sur WSL 2, QEMU. et KVM ci-dessous.

Mais pourquoi une si longue ligne de processeur ?

Aucun des types de processeurs d'origine dans QEMU ne correspondait bien à l'environnement WSL. J'ai donc commencé avec un kvm64 de base, puis j'ai rajouté tous les indicateurs de processeur pris en charge dans Hyper-V, puis j'ai testé pour voir lesquels des indicateurs vmx fonctionnaient. Les extensions VMX sont les extensions Intel VT-x pour accélérer les machines virtuelles.

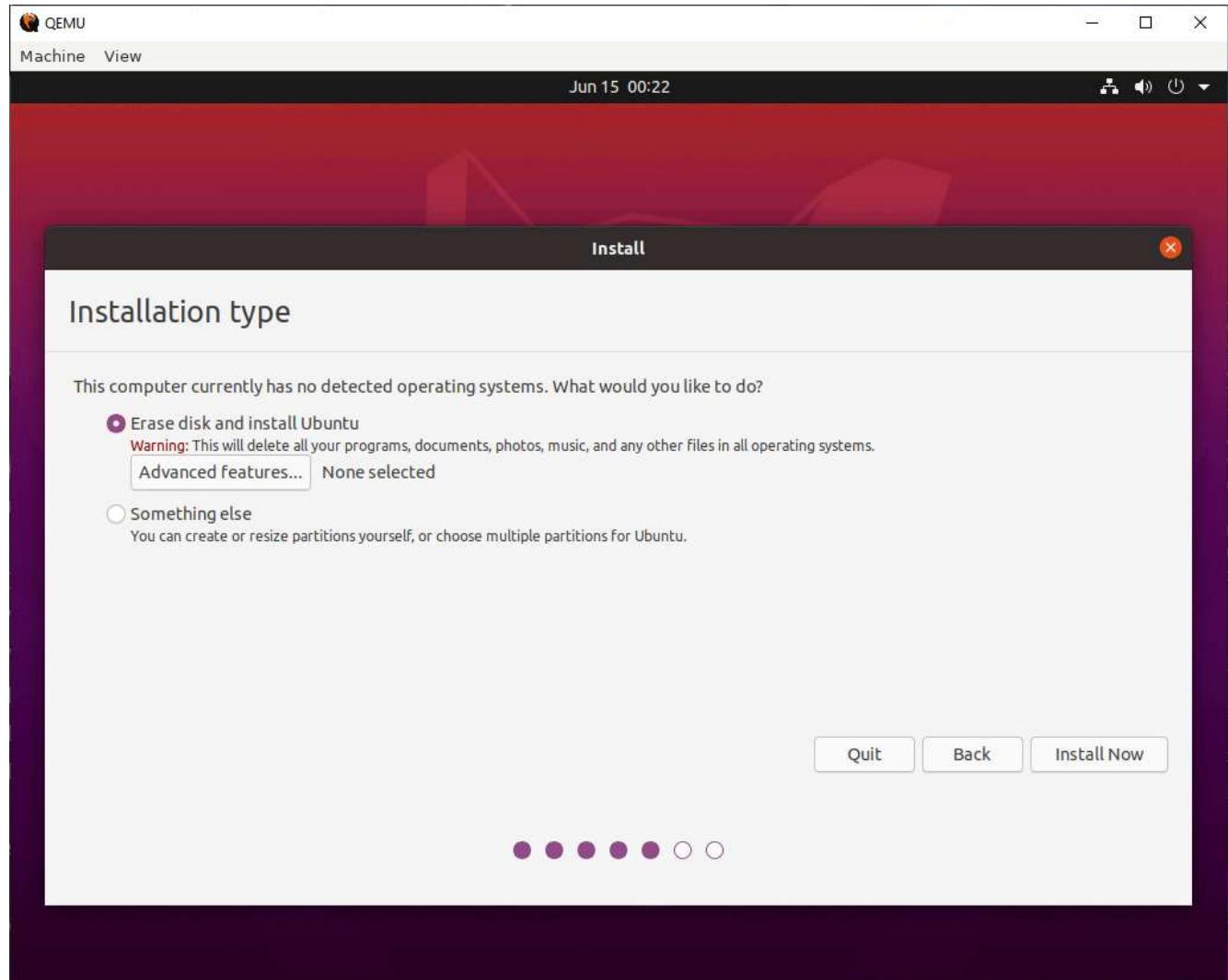
Après avoir peaufiné le noyau et exposé ces indicateurs de processeur, j'ai trouvé des améliorations significatives des performances de KVM sur Ubuntu WSL et, à ma grande surprise, la possibilité d'exécuter un KVM imbriqué sur imbriqué.

Futures bottes

Vous pouvez omettre la ligne ISO de démarrage :

```
drive file=groovy-desktop-arm64.iso,media=cdrom,readonly \
```

après avoir installé Ubuntu sur ubuntu.qcow2 :



Créer un script de démarrage

```
cd ~  
nano boot_ubuntu_kvm.sh
```

Pâte:

```
#!/bin/bash  
qemu-system-x86_64 \  
    -drive file=ubuntu.qcow2,format=qcow2 \  
    -net nic -net user \  
    -m 5172 \  
    -vga qxl \  
    --enable-kvm \  
    -smp 4 \  
    -cpu kvm64,+vmx,+vme,+msr,+x2apic,+hypervisor,+aes,+vmx-activity-hlt,+vmx-cr
```

Quittez et enregistrez.

Marquez le fichier comme exécutable :

```
sudo chmod u+x boot_ubuntu_kvm.sh
```

Maintenant, il peut être exécuté comme :

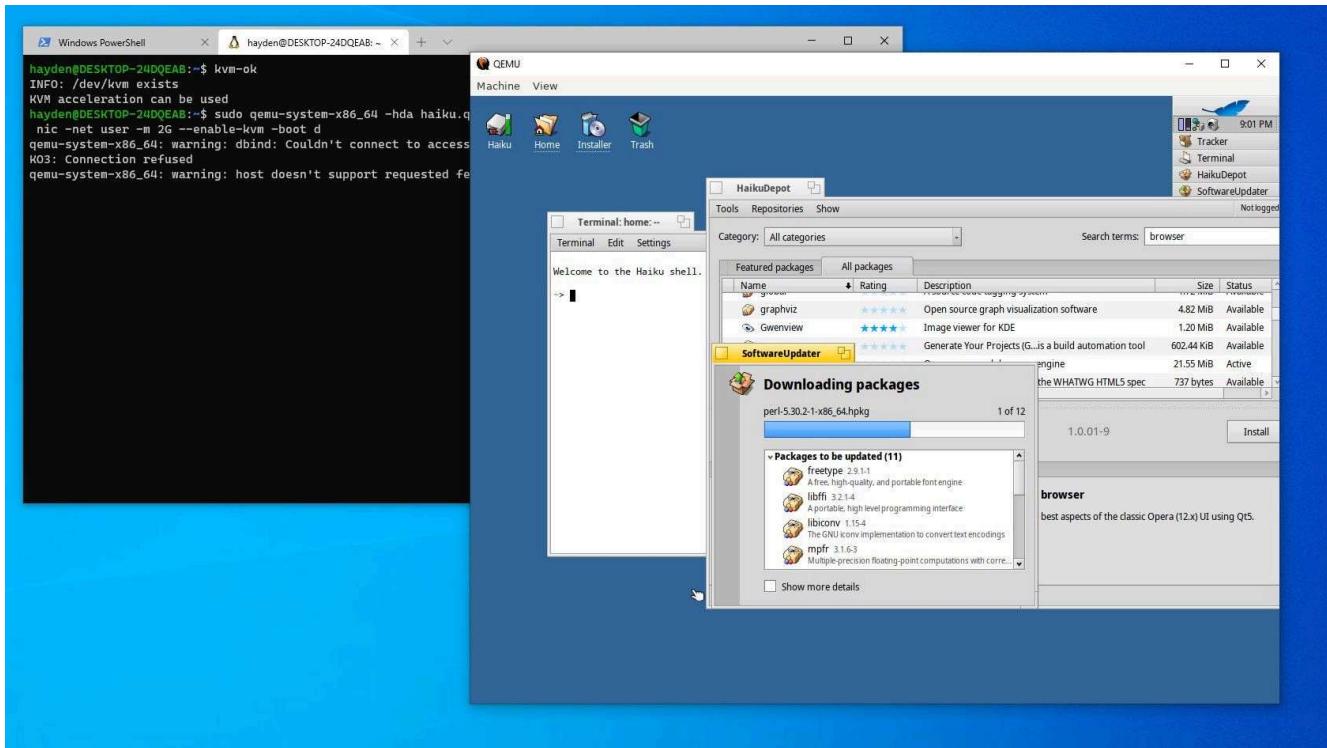
```
./boot_ubuntu_kvm.sh
```

Autres distributions

Vous pouvez créer autant de disques durs virtuels qcow2 différents que vous disposez d'espace et installer différentes distributions sur chacun.

Pour les autres distributions, créez simplement un nouveau fichier qcow2, téléchargez l'ISO amorçable (utilisez à nouveau aria pour l'accélérer, astuce : aria fonctionne également sur les fichiers .torrent !), montez les deux dans vos options qemu et installez-les sur le disque dur virtuel.

Autres systèmes d'exploitation



Haïku

- Téléchargez un ISO à partir des [images du soir](#).
- Suivez l'approche utilisée ci-dessus.

macOS

- Vous devez respecter les termes du CLUF macOS.
- Une fois la configuration KVM ci-dessus en place, [cela fonctionne](#) plutôt bien.

- Dans basic.sh, la tablette USB fonctionne mieux que la souris USB.

Lecture supplémentaire

monsieurbarbe/Génial-WSL

Liste impressionnante dédiée au sous-système Windows pour Linux - sirredbeard/Awesome-WSL

 GitHub • monsieurbarbe



Guide de l'utilisateur de l'émulation du système QEMU — Documentation QEMU 5.0.50 (v5.0.0-1244-g7d3660e798)

QEMU

QEMU — ArchWiki

 ArchWiki



Invités imbriqués - KVM

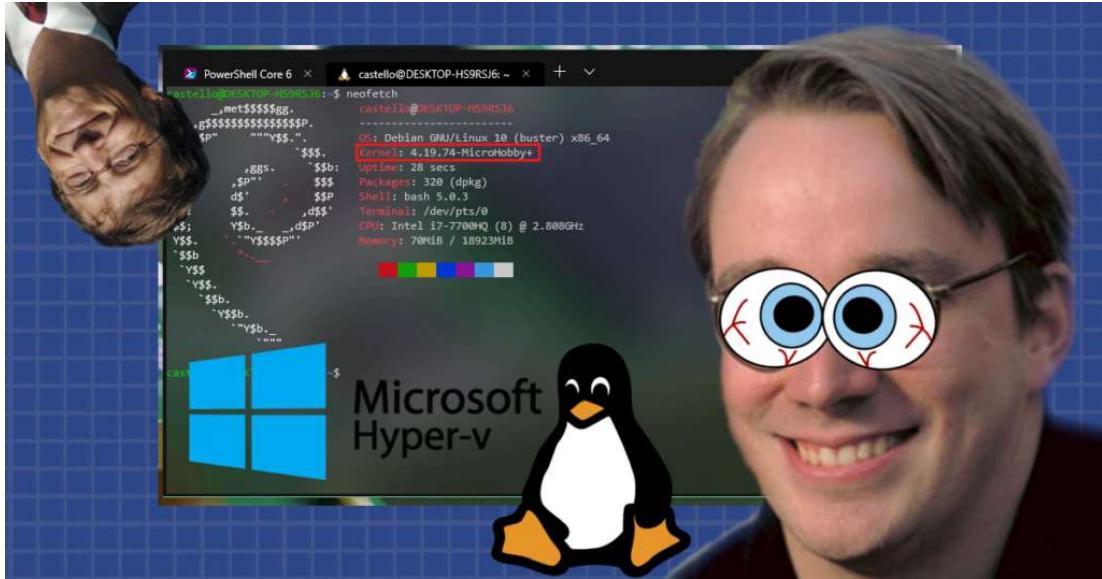
 KVM



WSL — Wiki Ubuntu

Wiki Ubuntu

[Compiler votre propre noyau Linux pour Windows ? WSL2](#)



Compiling Your Own Linux Kernel for Windows? WSL 2

But huh? What do you mean by compiling your own Linux Kernel for Windows!? Well little friends, if you haven't been in a cave in the last few months you've certainly heard that Microsoft will ... Continue reading

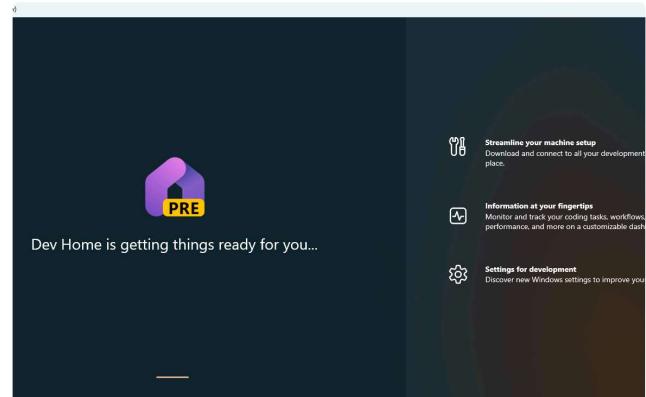
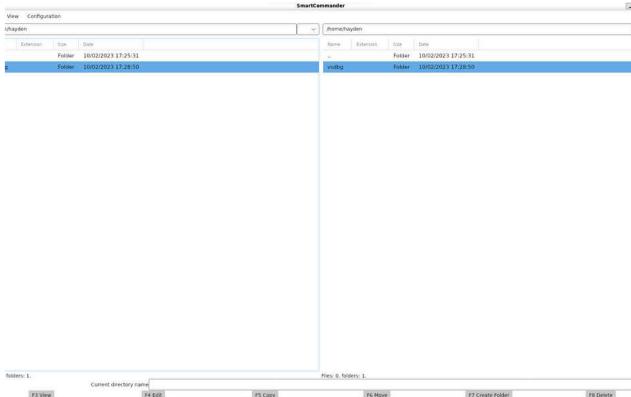


MicroHobby

Inscrivez-vous pour en savoir plus.

Entrer votre Email

S'abonner



Premiers pas Crédit d'une application de bureau Linux dans Visual Studio à l'aide d'Avalonia et WSL

Créez des applications GUI multiplateformes pour Windows et Linux à l'aide de C# et .NET à l'aide d'Avalonia da...



Hayden Barnes

2 octobre 2023 • 5 minutes de lecture

Benchmarks du lecteur de développement Windows

Mettre Windows Dev Drive à l'épreuve sur le Windows Dev Kit 2023.



Hayden Barnes

30 mai 2023 • 3 minutes de lecture

Boîte de câbles © 2024

Propulsé par Ghost